ATHABASCA UNIVERSITY

MODELLING MARKET BEHAVIOUR WITH ENSEMBLES AND TECHNICAL INDICATORS

BY

RYAN STEELE

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN INFORMATION SYSTEMS

FACULTY OF SCIENCE AND TECHNOLOGY
SCHOOL OF COMPUTING AND INFORMATION SYSTEMS

ATHABASCA UNIVERSITY
JANUARY, 2019

Approval of Thesis

The undersigned certify that they have read the thesis entitled

MODELLING MARKET BEHAVIOUR WITH ENSEMBLES AND TECHNICAL INDICATORS

Submitted by

Ryan Steele

In partial fulfillment of the requirements for the degree of

Master of Science in Information Systems

The thesis examination committee certifies that the thesis
and the oral examination is approved

Supervisor:
Dr. Larbi Esmahi
Athabasca University

Committee Members:
Dr. Dunwei Wen
Athabasca University

External Examiner:
Dr. Chunsheng Yang
National Research Council Canada

February 11, 2019

1 University Drive, Athabasca, AB, T9S 3A3 Canada
P: 780.675-6821 | Toll-free (CAN/U.S.) 1.800.788.9041 (ext 6821)
fgs@athabascau.ca | fgs.athabascau.ca | athabascau.ca

ACKNOWLEDGEMENTS

I would like to acknowledge my life partner, Stacey Charlton, who helped review early drafts of this thesis and provide feedback. Stacey also provided day-to-day emotional support and encouragement to bring this research to completion.

I would like to acknowledge my supervisor Dr. Larbi Esmahi of Athabasca University who has been supportive throughout my research efforts. I would also like to thank committee members Dr. Dunwei Wen and Dr. Chunsheng Yang. Both Dr. Esmahi and Dr. Wen provided constructive feedback and valuable suggestions to improve the design and presentation of this research.

I would like to acknowledge the financial support received from Alberta Advanced Education in the form of the Alberta Innovates – Technology Futures (AI-TF) Graduate Student Scholarship. The financial support provided by the AI-TF primarily allowed for the procurement of computer equipment used to conduct this research.

Lastly, I would like to acknowledge Shanaya Nelson of Amaya Editing Inc. who helped to format this document and provided initial copy-editing services. Shanaya provided suggestions to improve the presentation and readability of this thesis. Any errors in this thesis, whether content, formatting, grammatical or otherwise, are my own.

ABSTRACT

This research investigated the ability of different classification ensemble models to predict the outcome of market events defined by a technical trading system. The ensemble classification models used a diverse set of technical indicators to measure various aspects of market sentiment at the time of market entry as determined by a technical trading system. This research found that various ensemble classification models differ in their ability to classify the nonlinear relationships of market behaviour and are able to perform better than random chance in most cases.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF FREQUENTLY USED ABBREVIATIONS

ANOVA...................Analysis of Variance

BB............................Bollinger Bands

CART ....Classification and Regression Trees

CFS ....Correlation-Based Feature Selection

DT............................Decision Tree

ECOC ....Error Correcting Output Code

EMA ....Exponential Moving Average

EOD.........................End of Day

FNs ....False Negatives

FPs...........................False Positives

KNN .... K-Nearest Neighbour

MCC ....Mathews Correlation Coefficient

RUSBoost ....Random Under Sampling Boost

S&P ....Standard and Poor's Index

SMA ....Simple Moving Average

SMA2 ....Two Simple Moving Averages

SVM ....Support Vector Machine

TNs ....True Negatives

TPs...........................True Positives

Note: Appendix A contains a glossary of commonly used terms

CHAPTER I: SIGNIFICANCE OF THE PROBLEM

Background

Trading in the stock market relies on the increase (or decrease) in valuation of a stock based on supply and demand, business management, and company reputation. Statistical analysis and regression modelling have become popular techniques for forming expectations about the direction and magnitude of future financial security price movements. The more traditional and common modelling methods are built on the assumption of a linear relationship between predictor variables (features) and the response variable. Linear modelling methods do not account for the variability in nonlinear factors that interact to influence financial security prices such as idiosyncratic performance, investor psychology, limitations of available information, and politics.

For investment strategies with short-term horizons, researchers have stated that security prices correlate significantly with the reaction to market event information. Machine learning techniques have become common in nonlinear applications including modelling of security market pricing and market event information (Yoo, Kim, & Jan, 2005). Regression trees and K-nearest neighbour are examples of such supervised machine learners commonly used to model nonlinear relationships (Wu et al., 2008).

Speculative (short-term) investors may rely on nonlinear modelling methods, or machine learning techniques, to react to market news and take positions in anticipation that the financial security price will rise (or fall) in the near future. A short-term investor will evaluate variations in a security's price in an effort to identify the changes in the market equilibrium and the corresponding trend. The common objective of short-term investors is to accurately anticipate the next move of the market. This can be accomplished by “recognizing recurring patterns in price movement and determining the most likely results of such patterns [and] identifying the ‘trend’ of the market by isolating the basic direction of prices over a selected time interval” (Kaufman, 2013, p. 5). A unique aspect of this research is modelling a dynamic market *event*, which attempts to address both of these aspects.

Technical Trading and Machine Learning

Technical analysis is the systematic evaluation of past market data to estimate future price trends and make investment decisions (Kaufman, 2013, p. 1). The future investment horizon is unknown at the beginning of a trade, but it is shorter than a traditional passive investment strategy. Trading based on technical analysis predefines the conditions for both the entrance and exit of a market position instead of in accordance with a specific schedule. With predefined conditions that denote the beginning and end of a market position, the market for a security can be viewed as a series of events. Each market event is associated with a set of market conditions that cause the equilibrium of a security’s price to change.

The premise of this research suggested that market conditions can be modelled with the use of technical indicators and machine learning ensembles to classify the outcome of a market event. An ensemble is a collection of classifiers that combine the decisions of individual classifiers into a single predictor (Witten, Frank, & Hall, 2011, p. 352). Ensemble techniques have been proven, both theoretically and empirically, to outperform a single prediction model approach (Han, Kamber, & Pei, 2012, p. 377). This research was conducted to evaluate the performance of different ensemble classification models used to predict the outcome of market events defined by a technical trading system using various technical indicators to measure market sentiment, specifically the trend, momentum, volatility, and volume of a particular financial security.

Research Objective

Financial risk can be defined as “the uncertainty of future outcomes” (CFA Institute, 2008, Vol 4, p. 229). Traditionally, the risk of a financial security has been measured as the variance of returns relative to the mean return over some period of time (Mangram, 2013, p. 64).

Technical trading strategies focus on trends (or in the case of this research, events) rather than period returns. It is assumed that a target investment horizon of less than one quarter of a fiscal year focuses more on the market behaviour of a financial security rather than traditional fundamental analysis; therefore, modelling market events is reflective of the behaviour or how the market participants react to changing market conditions or recent news regarding a financial security.

This research focused on models that can be used to form expectations regarding the return and duration of a market event defined by a technical trading system. The focus of this research was to classify the outcome of a market event dynamically defined by a technical trading system using machine learning classification techniques. For the purposes of this research, the time between consecutive exit trading signals of a technical trading system are referred to as an event. Examining historical events allows technical traders to develop an expectation of future event returns, which is important to screen opportunities and allocate capital.

This research suggests that risk is relative based on the predictability of event outcomes, and the degree of competitive advantage offered by this research to a technical trader is proportional to the value of hidden information found in the market pricing. This research investigated if a trader who uses short-term technical trading strategies is able to gain an advantage by filtering trading opportunities identified by a particular trading system based on the predicted outcome label.

Research Questions

The researcher hypothesized that different ensemble classification models will have varying abilities to classify the outcome of events defined by short term technical trading strategies using technical indicators as measures of market sentiment. Furthermore, the researcher hypothesized the most favourable ensemble model will be more accurate than random chance. The following objectives and questions guided this research:

- Determine if there is a difference in the performance among the selected ensemble classification models in terms of identifying the outcome of an event.
- Determine the interaction/significance of the financial security and technical trading system in the performance of the ensemble classification models
    - Does the performance of the ensemble classification model (or models) vary depending on security or trading system?
    - Is there interaction between the security, trading system, and ensemble classification model (or models)?
    - To what degree does the selection of model features influence the performance of the ensemble model (or models)?

- Determine if it is possible to develop an ensemble classification model that is able to predict the outcome of an event using a diverse set of technical market indicators with a performance level that can be considered superior to random chance.
- Identify the technical indicators of trend, momentum, volatility and/or volume that are commonly identified as important predictors/features.

CHAPTER II: LITERATURE REVIEW

Background research was conducted to assess how fundamental analysis, market efficiency, and traditional time series forecasting techniques relate to the application of financial machine learning and behavioural finance. Literature that focuses on existing machine learning models was also reviewed to identify gaps within those models and how the proposed research can address these outstanding issues.

Fundamental Analysis

Fundamental analysis considers information contained on financial statements such as annual reports, balance sheets, and income statements. Fundamental analysis also investigates the relationship between macroeconomic data and the influence on returns of individual securities in addition to the market index as a whole. Fundamental analysis relies on the belief that every security has its intrinsic value, and if the share price is lower than the intrinsic value, the security is undervalued and vice versa (Tsai & Hsiao, 2010). The researcher argues fundamental analysis is applicable for target investment holding periods greater than 3 months or 1 quarter, which is the frequency that financial statements are generally issued. In contrast, this research primarily focused on target holding periods commonly less than 3 months.

Market Efficiency

Within an open market, market participants (buyers and sellers) determine the price of a financial security. The theoretical equilibrium price of a stock is where the return on investment (appreciation of share value plus dividends) balances with the risk of the investment relative to the returns of a risk-free alternative (Kaufman, 2013, p. 7). As the supply and demand dynamics of a security changes, the price must also change to bring the market back into equilibrium. More specifically, if the supply exceeds the demand, the security price should fall, and if the demand exceeds the supply, the security price should rise (Tsai & Hsiao, 2010). Security prices will seek a level that balances the supply and demand factors.

Market efficiency refers to the speed that information is reflected in security pricing (Gold & Lebowitz, 1999). In a perfectly efficient market, new information concerning a security should be reflected instantaneously. However, if only some of the information is reflected in the security pricing instantaneously, and the remaining information takes a number of periods to be reflected (hours, days, or weeks), then the market is less than fully efficient, leading to short-term mispricing and an opportunity to generate excess profit (CFA Institute, 2008, pp. 95–96). Markets cannot be fully efficient because of the costs associated with collecting and analyzing information, the cost of trading, and the limit of available capital to arbitrageurs (CFA Institute, 2008, p. 109). As a result, security pricing generally reflects information up to the point at which marginal benefits equal the marginal costs of information (Gold & Lebowitz, 1999). This research did not directly target market efficiency; however, the effects of market efficiency may indirectly interact with the technical trading systems and their defined entry and exit triggers.

Traditional Time Series Forecasting Techniques

The most common financial time series forecasting approaches are auto regressive integrated moving averages (ARIMA) models, or simpler visions of this approach such as auto regressive moving averages (ARMA) models. In addition, generalized autoregressive conditional heteroskedasticity (GARCH) models have been historically used to estimate variance. An ARIMA and/or GARCH model is used to predict future price movements in cases in which financial time series data show evidence of non-stationary/time varying volatility (Preethi & Santhi, 2012, p. 28). An in-depth discussion of ARIMA and GARCH was beyond the scope of this research work. This research used a variety of time-dimensioned features to capture non-stationary characteristics and focused on machine learning rather than traditional forecasting approaches. Readers interested in ARIMA and GARCH are encouraged to reference the textbook *Elements of Forecasting* by Diebold (2004).

Fang and Xu (2003) investigated an approach that combines technical analysis and traditional time series forecasts. This research built on Fang and Xu's idea that technical trading rules and time series forecasts capture different aspects of the market and went further to argue that the historical time series of a financial security should be viewed as a series of discrete events, with each event representing a change in price equilibrium of the security.

Behavioural Finance

Market conditions and general market sentiments are based on the behaviour of market participants, which is an extension of human behaviour. When most market participants hold similar expectations regarding a security, prices move quickly to align with the common expectation (Kaufman, 2013, p. 7). Many professional investors have argued that financial markets are dominated by the emotions of fear and greed that lead to investors addressing risky choices based on emotion instead of fact (Goedhart, Koller & Wessels, 2005). Some investors are reluctant to realize losses, take profits too quickly, or suffer from other instances of irrational behaviour, all of which can lead to mispricing in the market (CFA Institute, 2008, p. 106).

Most importantly, investors do not appear to be consistent in how they treat economically equivalent choices if those choices are presented in different contexts. Authors have suggested framing effects to have an impact on rational decision making (Sharpe, Alexander, & Bailey, 1999, p. 146). Market participants tend to overestimate the probability of unlikely events occurring and underestimate the probability of moderately likely events occurring, causing the tendency of investors to overreact to good and bad news (Bodie, Zane, & Marcus, 2003, p. 178). This notion can be supported by the historic "'rushes', 'booms', 'busts', [and] 'bubbles'" (Mangram, 2013, p. 67), which provide creed to the notion that markets are less than perfectly efficient. A number of researchers stated that stock prices are significantly correlated with the reaction to market event information (Yoo et al., 2005).

Machine Learning and Financial Applications

Historically, investors have used regression and statistical analysis for predicting the direction of market pricing. The most common modelling methods are variations of econometrics, basic probability theory and statistics (Kaufman, 2013, p. 6). Unfortunately, many real-world problems are not simple linear projections of previous values (Imandoust & Bolandraftar, 2013). Although analysts have widely used multivariate models for predicting security pricing and stock market movements, several machine learning techniques are now becoming more common (Yoo et al., 2005).

Neural networks (NNs) have received the most attention in existing research related to forecasting security price movements; Tsai, Lin, Yen, & Chen (2011) have suggested NN outperform many other statistical based techniques such as regression and discriminant analysis. NNs have the ability to extract useful information from large sets of data (Preethi & Santhi, 2012, p. 24). Over the last decade, NNs have shown to better estimate future security returns over some traditional approaches, but they have the tendency to overfit the data or find a local minima solution (Tsai & Wang, 2009; Yoo et al., 2005).

In terms of application, Tsai and Wang (2009) proposed a security price forecasting model that combines an NN with a decision tree to address the potential of overfitting. Weckman and Agarwala (2003) investigated the sensitivity of different technical indicators in an artificial neural network (ANN), which was used to forecast a security *n*-periods into the future. Case-based reasoning (CBR) and support vector machines (SVM) were also reviewed. CBR can reuse past cases (or historic market positions) to estimate the outcome of new opportunities (Yoo et al., 2005). SVM models are based on statistical learning theory and are able to find a globally optimal solution (Leung, MacKinnon, & Wang, 2014; Yoo et al., 2005). This research recognized the effectiveness of SVM models and relied on an SVM model as a benchmark for other machine learning ensemble classification models.

Selected Related Research

The following sections provide a summary of focused research specific to machine learning and related to the research objectives of this thesis. The topics discussed include the prediction period (forecast horizon), the use of technical indicators, model performance comparisons, machine learning and trading systems combinations, market behavior prediction, and the use of complex trading rules to forecast direction.

Prediction period

Krollner, Vanstone, and Finnie (2010) completed a literature survey in the domain of machine learning techniques and artificial intelligence used to forecast stock market movements. The prediction periods are categorized into 1-day, 1-week, and 1-month ahead predictions. Most papers make 1-day ahead predictions (e.g., predicting the next day's closing price). However, being able to predict the stock index 1-day ahead may limit how an investor can take advantage of this information in terms of trading profit. This thesis proposed the concept of an event, which attempts to frame common market behaviour based on indicator trends and not a specific fixed length prediction period. Various technical indicators in this research are used to represent the non-stationarity of market sentiment and classify the outcome of the event.

Technical indicators

Krollner et al. (2010) showed that over 75% of the reviewed papers rely in some form on lagged index data. The most commonly used parameters are daily opening, high, low, and close prices. Krollner et al. (2010) further found the most common technical indicators identified in their surveyed literature were the simple moving average (SMA), exponential moving average (EMA), relative strength index (RSI), rate of change (ROC), moving average convergence/divergence (MACD), William's oscillator, and average true range (ATR).

Machine learning techniques commonly employed are existing ANN models enhanced with new training algorithms or combined with emerging technologies into hybrid systems. Lagged index data and derived technical indicators have been identified as the most popular input parameters in the literature. The purpose of this thesis was to determine if it is possible to gain a competitive advantage using only financial security pricing data; therefore, this research also used technical indicators as input parameters.

Machine learning approaches comparisons

Ahmed, Atiya, Gayar, and El-Shishiny (2010) presented a comparison among eight machine learning models. The models considered by Ahmed et al.'s study were multilayer perceptron, Bayesian NNs, radial basis functions, generalized regression NNs (also called kernel regression), K-nearest neighbour (KNN) regression, classification and regression trees (CART), support vector regression, and Gaussian processes. The machine learning models were considered in their basic forms without the modifications and the additions proposed by other researchers. Ahmed et al.'s study revealed significant differences between the different methods and identified the multilayer perceptron and the Gaussian process regression methods as preferred methods for accuracy and precision. In addition to model comparisons, the authors tested different preprocessing methods and have shown that they have different impacts on the performance (Ahmed et al., 2010).

Tay and Cao (2001) examined the feasibility of SVM in financial time series forecasting by comparing it with a multi-layer back-propagation (BP) NN. The objective of Tay and Cao's paper was to examine the feasibility of applying SVM in financial forecasting by comparing to a BP NN, and, secondly, to investigate the functional characteristics of SVMs in financial forecasting. The functional characteristics were obtained through the selection of the free parameters of SVMs. Using five futures contracts from the Chicago Mercantile Market as data, Tay and Cao showed that SVM outperforms the BP NN based on the criteria of normalized mean square error (NMSE), mean absolute error (MAE), directional symmetry (DS), and weighted directional symmetry (WDS), concluding that it is advantageous to apply SVMs to forecast financial time series.

This thesis considered different machine learning ensemble models in addition to an SVM as a benchmark comparison. The machine learning approaches included in this research is focused on ensembles, specifically bagging (random forest), boosting (random under sampling), and subspace (K-nearest neighbour). These machine learning ensemble models were chosen for their ability to perform both binary and multi-class classification.

Machine learning and trading system combinations

R. Dash and Dash (2016) proposed a decision support system using a computational efficient functional link artificial neural network (CEFLANN) and a set of rules to generate trading decisions. The decision prediction was expressed as a classification problem with three class values representing a buy, hold, and sell signal.

In R. Dash and Dash’s (2016) study, six popular technical indicators were chosen as inputs. The six technical indicator values represented continuous values. The input data were scaled in the range 0–1 using the min-max normalization. Scaling the input data ensured that larger value input attributes did not overwhelm smaller value inputs. The evaluation of the model performance was measured through a 5-fold cross-validation approach applied on the initial 1,000 samples taken as training data set. The training data set was divided into five groups, in which the first four randomly chosen groups were used for training and the remaining fifth group was used for validation. The authors considered the average performance out of the 20 independent runs (Dash & Dash, 2016).

Finally, the authors applied the trained network on the test pattern (i.e., the out of sample data), which had not been used during training and validation (Dash & Dash, 2016).

R. Dash and Dash (2016) compared their model performance to other machine learning techniques such as SVM, naïve Bayesian model, K-nearest neighbour (KNN) model and decision tree (DT) model. From the experimental results, R. Dash and Dash demonstrated that their proposed model provided a greater profit percentage compared to the other models listed above. Relating to this research, R. Dash and Dash (2016) concluded it is more profitable to frame trading decisions using a combination of technical indicators with computational intelligence tools than use any one particular technical indicator as a decision system. R. Dash and Dash suggested future research should include validating the proposed model over more real-world data sets and exploring more technical analysis. This thesis did not evaluate the optimal parameters of the trading system; rather, this thesis endeavoured to classify the outcome of an event defined by a common trading system and filter out less favourable opportunities.

Sezer, Ozbayoglu, and Dogdu (2017) proposed a stock trading system based on optimized technical analysis parameters for creating buy-sell points using a genetic algorithm. The authors calculated SMA values to determine whether the trend is up or down and the relative strength index (RSI) indicator was used for buy-sell points (Sezer et al., 2017). Using a genetic algorithm, Sezer et al. determined the best RSI values for buy and sell points during downtrend and uptrend. Their results indicated that optimizing the technical indicator RSI parameters enhanced the stock trading performance (Sezer et al., 2017). Their results also indicated that their trading system produces comparable or better results when compared with the buy and hold strategy and other trading systems for a wide range of stocks.

Sezer et al. (2017) used 30 stocks from the Dow Jones for model validation. Each stock was trained separately using daily close prices between 1996–2006 and tested between 2007–2016. The authors obtained Dow 30 stock price data from finance.yahoo.com between January 1, 1997, and December 31, 2006 and between January 1, 2007, and January 1, 2017, for training and testing purposes, respectively (Sezer et al., 2017). The authors first normalized the downloaded stock prices according to the adjusted close prices (Sezer et al., 2017).

Sezer et al. (2017) focused on optimizing the trading system component. In contrast, this thesis investigated the competitive advantage of classifying the general outcome and direction of an event that is defined by a common trading system and focused on the use of ensemble models, rather than a genetic algorithm.

Behaviour prediction

Wang, Xu, and Zheng (2018) proposed an approach that uses both social media and market technical indicators in stock market prediction. The effects of textual information provided by news articles on stock price movements are included in their model (Wang et al., 2018). These authors developed a data-mining technique called deep random subspace ensembles (DRSE) that integrates deep learning algorithms and ensemble learning methods for more effective mining of stock market fluctuations (Wang et al., 2018).

Wang et al. (2018) viewed stock market prediction as a binary classification problem. Using social media sentiment and market technical indicators, they developed an approach that predicts stock market fluctuations based on data-mining techniques (Wang et al., 2018). For the collected data set, each sample was supplemented with a fluctuation flag that indicates whether the stock market will go up or down the next day. The whole data set was divided into a training set and a test set. After the deep learning classifiers were trained, the predictive model labeled the fluctuation flags of samples from the test set. They employed a 10-fold cross-validation method for model evaluation (Wang et al., 2018).

Wang et al. (2018) found comparing the experimental results of the same classifiers under different feature sets and the addition of textual information in combination with technical features did help to improve the prediction accuracy of the stock market trend. In contrast, this thesis attempted to find patterns related to event outcomes. The intention was to evaluate if it is possible to predict behaviour identified using only security pricing data.

Complex trading rules to forecast direction

While other related studies try to accurately predict future price levels, Van den Poel, Chesterman, Koppenm, and Ballings (2016) focused on forecasting stock price direction. They investigated whether a two-layered model trained on an 8-month sample of minute-by-minute Standard and Poor's (S&P) 100 data is able to generate profits (Van den Poel et al., 2016).

Ten of the most popular technical analysis indicators were included in Van den Poel et al.'s (2016) research. The feature selection layer of the model consisted of 193 predictors used as inputs to the prediction layer of the developed model. In order to prevent features with larger values from overwhelming those with smaller values, the authors standardized the predictors such that the obtained indicator values were normalized into new scores with a mean of zero and standard deviation of 1 (Van den Poel et al., 2016). All technical trading rules were transformed to binary variables.

Van den Poel et al. (2016) showed that random forest ensemble classifier models have predictive power and yield better returns than the buy-and-hold strategy when disregarding transaction costs both in terms of number of stocks with profitable trades as well as overall returns. Furthermore, these authors showed that continuous-valued technical indicators add the most to the accuracy (Van den Poel et al., 2016). Complex trading rules outperformed single trading rules for both the five-indicator analysis and the ten-indicator analysis. On average, two-way combinations improved accuracy by more than 15%, adding a third rule increased the accuracy by an additional 7% on top of the 15%. Van den Poel et al. (2016) concluded that two-way and three-way combinations, (i.e., complex trading rules) are important to “beat” the buy-and-hold strategy.

The approach taken by Van den Poel et al. (2016) and this thesis share some commonality. This research was built on related works by introducing event outcome clustering, a multi-class classification approach with ensemble models, and dimensionality of the predictor variables to address the non-stationary properties of time series data.

The purpose of this research was to address a number of the outstanding issues relating to whether it is possible to gain a competitive advantage from using only security pricing data to train machine learning ensemble methods and predict with an increased accuracy better than random chance. Based on this literature review, three technical trading systems were selected in combination with three machine learning ensemble methods and one benchmark model to test the proposed hypotheses.

CHAPTER III: THEORETICAL FRAMEWORK

This section provides a theoretical framework for the research by summarizing the key theories and providing definition to terms that are used throughout this document. Summarized definitions of terms are also available for reference in the *Glossary of Terms* found in Appendix A.

Financial Security

A financial security is a tradable asset such as a bond (debenture), stock (equity), or other financial asset that has a time series of historical data. For the purposes of this research, a financial security refers to a randomly selected stock from the S&P 1500 Composite Index.

A symbol (or ticker) uniquely identifies a financial security. The symbol of a financial security can be used to fetch time series data from a market database. A financial time series has a periodicity. In the context of this research, periodicity refers to the frequency of observation (i.e., minute, hour, day, week, or month). The periodicity of the data used in this research was daily, specifically, end of day data. Periodicity is sometimes referred to as bar size.

The price bar shows a summary of the financial security's values within a period. The common elements of a price bar are open, high, low, and close (see *Figure 1*). Volume is also an element of financial security time series data and relates to the quantity traded within a period. The price bar is the basic building block of technical analysis.

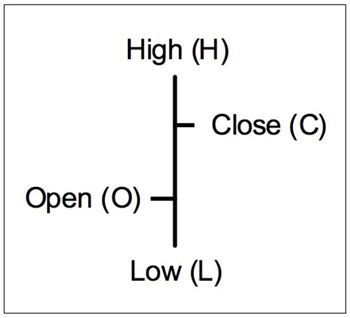


Figure 1. The common elements of a price bar.

Technical Trading Systems

The equilibrium price point of a financial security moves in response to the latest available information or news release (Kaufman, 2013, p. 6). The principle of a technical trading system is to identify the market behaviour of the traders. A technical trader wants to be sensitive to market activity without succumbing to the emotions of the market itself (Rockefeller, 2004, p. 25). The objective of technical analysis is to study what market participants do (price and volume), not what market participants say (Rockefeller, 2004, p. 26). The use of indicators in technical analysis allows for this distinction.

Technical indicators impose discipline on a technical trader's trading practices as well as clarify the perceptions of a price movement bypassing greed, fear, and other emotions that accompany trading. The term *technical indicator* refers to a statistic that is derived from the price and/or volume data of a financial security (Achelis, 2001). It is the researcher's position that the purpose of technical analysis is to reorganize pricing data in a manner that provides insight about the market sentiment toward a financial security.

One of the critical elements of most technical indicators (and trading systems) is the number of time periods (or observations) used in the calculation. There are various popular values for the time parameter *n*, such as 5 days (more generically, periods), 10 days, 21 days, and so forth. The length of the period determines the nature of the underlying trend that will be targeted (Kaufman, 2013, p. 330). This parameter is referred to as the *time dimension* within this research. Various technical indicators with various time dimensions are used in this research to form the technical trading systems as well as to provide a collection of predictor variables to quantify market sentiment.

One or more technical indicators can be used to create a technical trading system. A technical trading system provides dynamic signalling of the entry and exit points for a market position. A notable portion of other literature reviewed focuses on optimizing the trading system. In contrast, this research work is aimed to model the behaviour of common trading systems. This research considered three common technical trading systems to confirm if historic market patterns are repeated: simple moving average (SMA), two simple moving averages (SMA2), and Bollinger bands (BB). Each of these technical trading systems is summarized in the subsections that follow.

Trading system I: Simple moving average

A moving average shows the typical value of a financial security's price over a period of time. As a financial security's price changes, the average price moves up or down accordingly. A moving average can be used to smooth short-term fluctuations of security pricing and highlight longer-term trends or cycles (Preethi & Santhi, 2012, p. 28). The differentiation between short- and long-term trends depends on the time dimension parameters of the moving average.

During a period when the market of a financial security is trending, moving averages can be effective in timing market position entry and exit signals (Kaufman, 2013, p. 287). The SMA is defined as the equally weighted mean of the most recent *n*-observations and can be calculated as follows:

$$SMA_t = \frac{P_{t-n+1} + P_{t-n+2} + \cdots + P_t}{n}$$

A *crossover* is a basic trading signal of a SMA trading system. Price crossovers are used by traders to identify shifts in momentum and can be employed as a basic entry or exit strategy. In the case of an SMA system, when the closing price crosses above a moving average price, it suggests the beginning of a new upward trend, and when the closing price crosses below a moving average it suggests the start of a downward trend (see *Figure 2*).

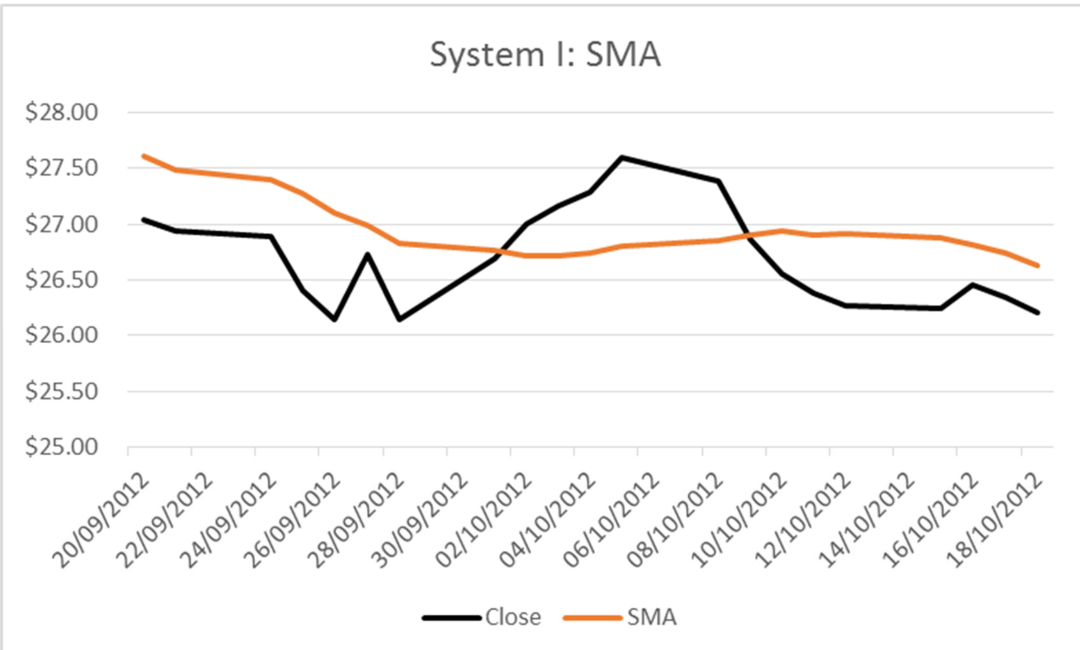


Figure 2. Simple moving average trend example.
Note. SMA = Simple Moving Average.

Trading system II: Two simple moving averages

The concept of a two simple moving average (SMA2) crossover trading system is similar in nature to a single SMA trading system. Two moving averages of a financial time series are created with one moving average having a shorter time dimension parameter than the other (e.g., 5-day period vs. 20-day period). The SMA with the longer time dimension will have a lower variance and will move more slowly in the same direction as the short-term moving average but at a different rate. In a SMA2 trading system, when the short-term moving average crosses above the long-term moving average, a buy signal is generated. Conversely, when the short-term moving average crosses below the long-term moving average, the system provides a sell signal (see *Figure 3*).

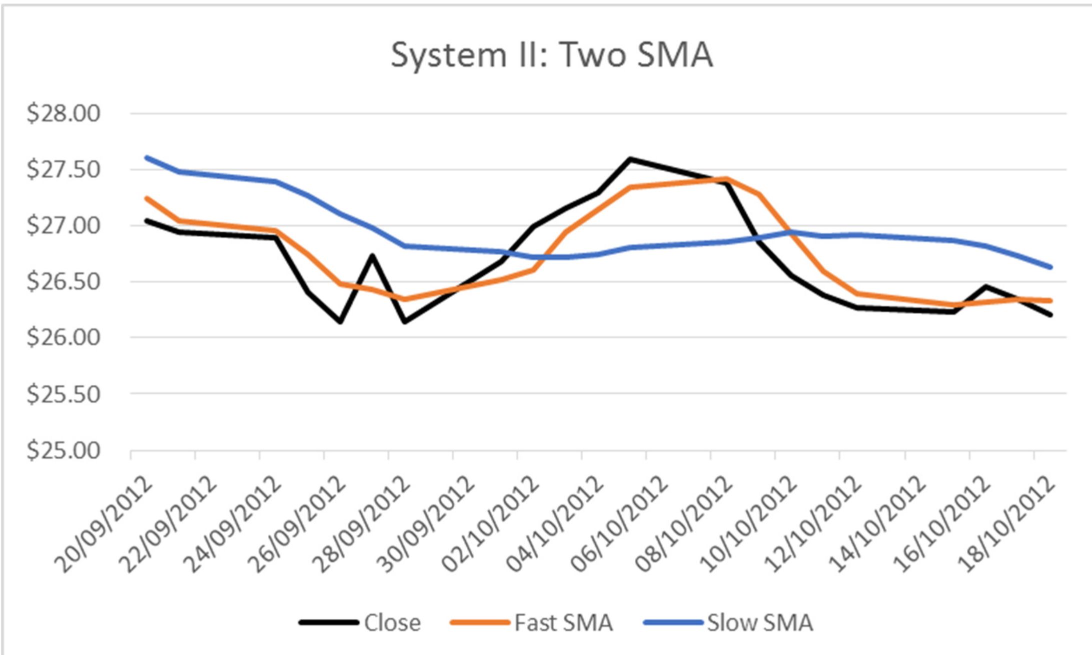


Figure 3. Two simple moving average trend example.
Note. SMA = Simple Moving Average.

Trading system III: Bollinger bands

Bollinger bands (BB) are a technical analysis tool invented by John Bollinger in the 1980s (Kaufman, 2013). The purpose of BB is to provide a relative definition of high and low, in which prices are considered relatively high at the upper band and relatively low at the lower band (Kaufman, 2013, p. 328). The BB tool consists of the following elements:

- A middle band being an *n*-period SMA.

- An upper band at K times an n-period standard deviation above the middle band (SMA + $K\sigma$).
- A lower band at K times an n-period standard deviation below the middle band (SMA − $K\sigma$).
- Typical values for n and K are 20 and 2, respectively. Usually, the same period is used for both the middle band and the calculation of standard deviation, where SMA refers to Simple Moving Average and σ is the standard deviation of the financial timeseries.

A BB system measures price volatility (as a function of standard deviation) and adjusts to market conditions. When the bands move closer together, constricting towards the moving average, the market for the security is experiencing a period of low volatility.

In contrast to an SMA system, BB are a counter-trend system (see *Figure 4*). Some market participants believe that the closer the prices move to the upper band, the more overbought the security becomes; therefore, the security prices should begin to move towards the lower band. As prices approach the lower band, the security is viewed to be oversold and prices should begin to rise.

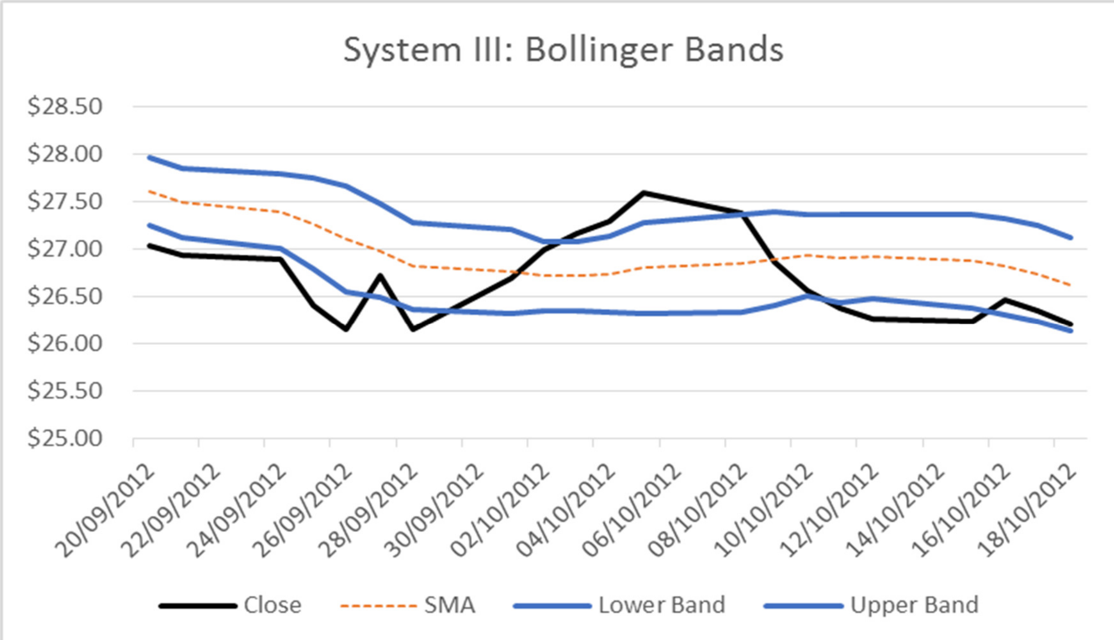


Figure 4. Bollinger bands trend example.
Note. SMA = Simple Moving Average.

This research defined trading rules for each of the previously discussed trading systems, which are presented in Table 1. Entering a long position occurs when a trader buys a stock with the intent to sell the stock at a higher price. Exiting a long position is when the trader sells the stock.

Table 1
Trading Rules Applied to Trading Systems

Trading System	Trading Rules
Simple Moving Average Crossover (SMA)	• Enter a long position when the closing price crosses above the moving average. • Exit the long position when the closing price crosses below the moving average.

Two Simple Moving Average Crossover (SMA2)	• Enter a long position when the fast SMA crosses above the slow SMA. • Exit a long position when the fast SMA crosses below the slow SMA.
Bollinger Bands (BB)	• Enter a long position when the daily closing price crosses the lower band. • Exit a long position when the daily closing price crosses above the moving average.

Note. SMA = Simple Moving Average; SMA2 = Two Simple Moving Averages.

Concept of an Event

A core element of this research is delineating the time series data of a financial security into a collection of market events. An *event* within this research is a dynamic cross-section of time defined by the entry and exit signals of a technical trading system. An event has three states (monitoring, active, and historic) defined by three discrete actions (spawn, start, end), as shown *Figure 5*.

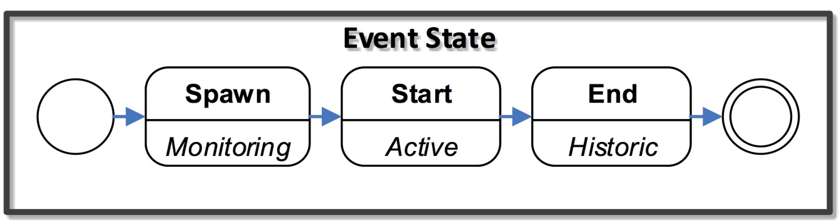


Figure 5. The three states of an event.

An event is spawn (created) in a monitoring state, waiting for an entry signal from the trading system (start). Once the trading system triggers an entry signal the event turns from a monitoring state to an active state representing the beginning of a market position. The event remains in an active state until the trading system triggers an exit signal (end). Once an exit signal is received, the event state is changed to historic, and a new event is spawned into a monitoring state, allowing the cycle to continue. This results in the time series of a financial security being delineated into a collection of discrete historic events and one current event. *Figure 6* shows the relationship between the three states of a financial security's time series separated into discrete events.

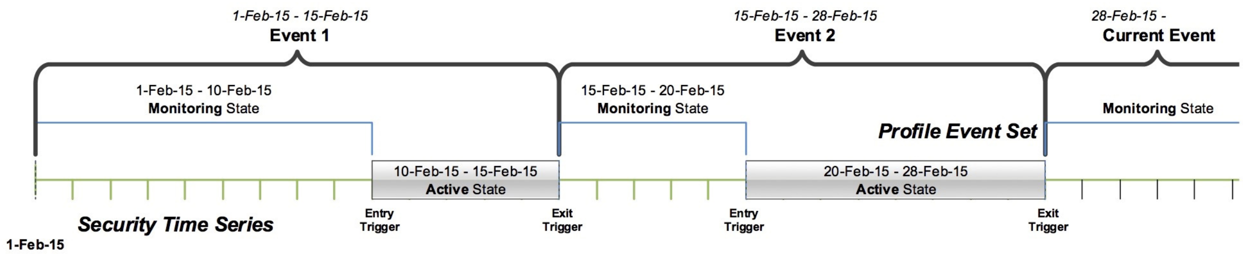


Figure 6. The relationship of discrete events and the three states of an event.

The concept of an event is rationalized as containing a price movement, which represents a change in equilibrium price for a given security. The objective is not to forecast the security pricing *n*-periods into the future, but to estimate the generalized result of a market event defined by the technical trading system. The generalized result is a classification label reflective of the event outcome with labels representing the degree of either a favourable outcome (profitable) or unfavourable outcome (loss).

Event outcome predictor variables: Candidate features

The classification models contained in this research use technical indicators as predictor variables to forecast the outcome classification label of an event. The technical indicators available to be used as predictor variables are referred to as *candidate features*. Each candidate feature is a composition of four dimensions, namely factor, technical indicator, time, and derivative. There are 1,596+1 candidate features generated at the time an event transitions from a monitoring state to an active state, which occurs when an entry signal is triggered. One of the candidate features, with the plus 1 representing the number of periods (days) the event was in a monitoring state. *Figure 7* provides a visual representation of the dimensions of the set of the remaining 1,596 candidate features.

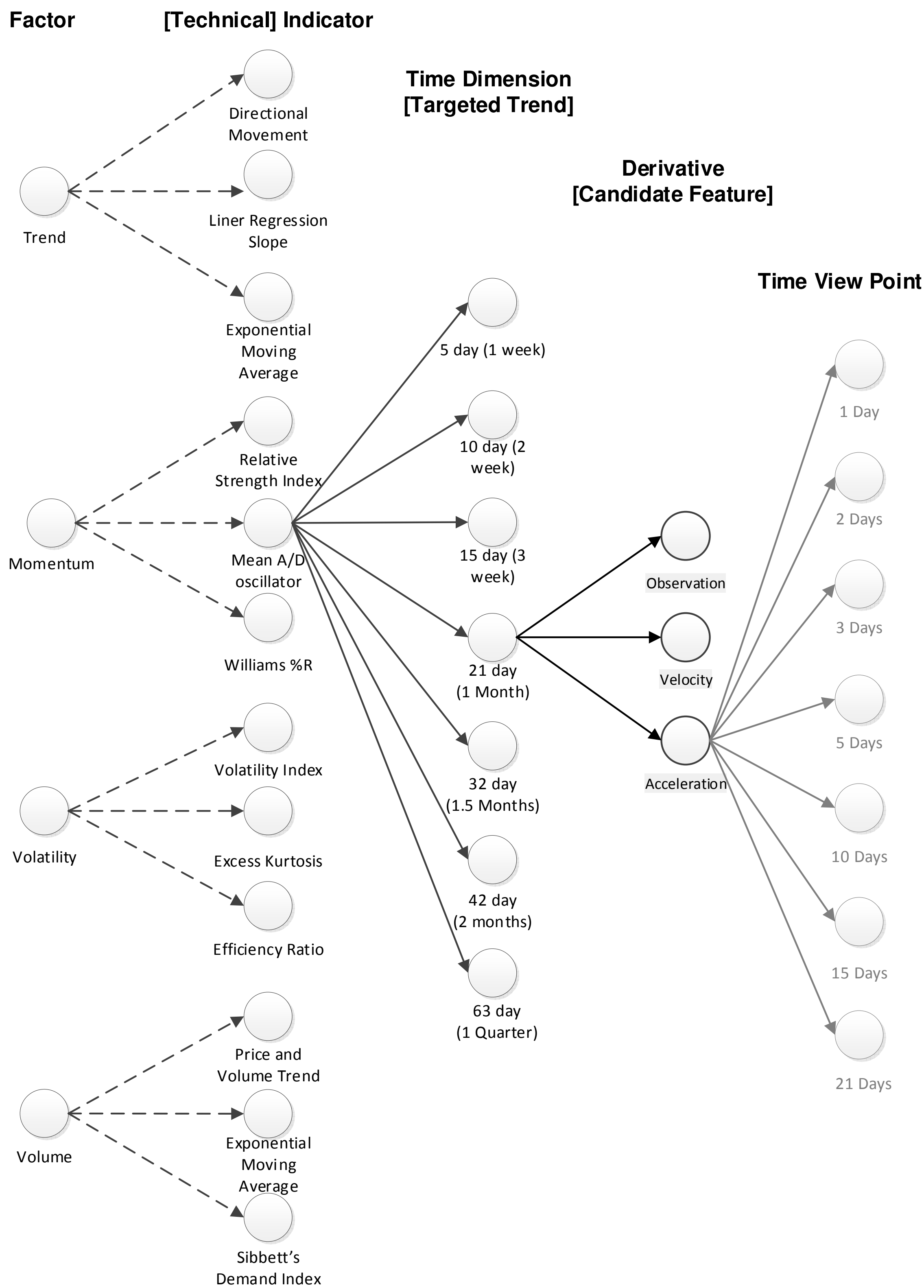


Figure 7. Candidate feature dimensions.

Dimension I: Factor

A factor is an aspect of market sentiment that is measured by a technical indicator. There are four identified factors, namely the strength of the *trend*, the *momentum* of the financial security price, the *volatility* of the security price, and the trading *volume*. In addition, the *monitoring duration* of the event, which is the number of periods since the conclusion of the last event, is considered an independent factor that has no associated indicators or other derivative measures.

Dimension II: Technical indicator

A technical indicator is a metric derived from the financial security's pricing and/or volume. There are 12 discrete indicators, three indicators for each of the four aspects of market sentiment. The details of each technical indicator, including its calculation, are presented in Appendix B.

Dimension III: Time

The time dimension is the parameter that specifies the number of price/volume observations to be included in the calculation of the technical indicator. Smaller values of *n*-observations increase the sensitivity of the indicator to recent market activity while larger values of *n*-observations decrease the sensitivity and smooth out market fluctuations. Each technical indicator is calculated based on the number of trading days in a year (252) over several predetermined time dimensions, which are presented in Table 2.

Table 2

Predetermined Time Dimensions of technical indicators

Time Dimension	Calculation	Predetermined Period in Days
1 week	252 trading days/ (52/1)	5 days
2 weeks	252 trading days/ (52/2)	10 days
3 weeks	252 trading days/ (52/3)	15 days
1 month	252 trading days/ (12/1)	21 days
1.5 months	252 trading days/ (12/1.5)	32 days
2 months	252 trading days/ (12/2)	42 days
1 quarter	252 trading days/ (12/3)	63 days

Dimension IV: Derivative

Each technical indicator has an observed value (d_t) and two derivatives, namely velocity (v_{ave}) and acceleration (a). These derivatives capture the rate of change of the technical indicators over time. The use of these derivatives in combination with the time dimension is intended to account for the non-stationary characteristics, auto correlation, and heteroscedasticity present in financial time series data.

The *view point* is the parameter used to calculate the derivative between technical analysis observations (TA Obs d_t). For example, velocity requires two points in time. The derivative measure is calculated as the difference in the indicator value at the time of an entry trigger and the indicator value n-periods earlier. Each derivative is calculated with predetermined view points for n periods equal to 1, 2, 3, 5, 10, 15, and 21. *Figure 8* depicts the relationships between the derivatives.

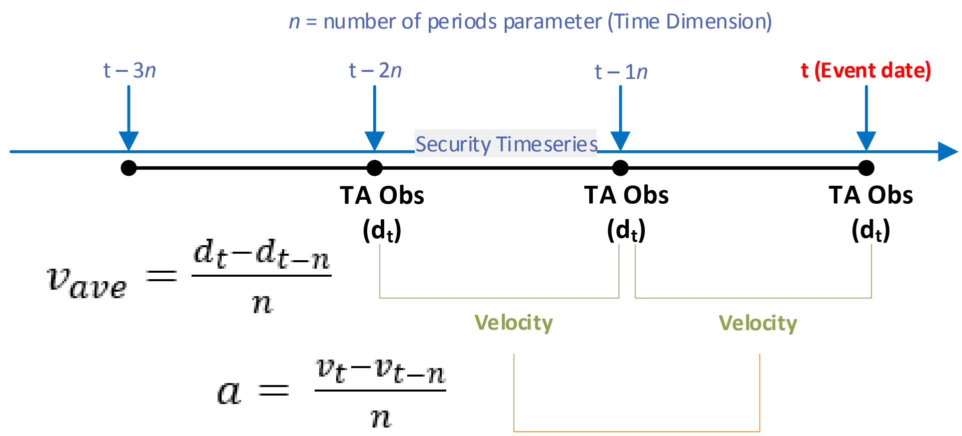


Figure 8. Derivative relationship.

Note: TA Obs (dt) = Technical Analysis Observation of derivative at time view *t*;

V_{ave} = Average Velocity; *a* = acceleration

Response Variable Modes

This research considers two different approaches to grouping the outcomes of events into classification labels, namely binning and clustering. Furthermore, this research considers both multi-class (multi) and binary classification.

Binning

Data binning is a method to group values into a smaller number of discrete bins. The bounds of the bins are predetermined and observations that fall within the bounds of a specific bin are assigned the corresponding bin label (CFA Institute, 2008, pp. 244-245).

K-means clustering

K-means clustering is an unsupervised learning algorithm that is used to partition data into K distinct groups (clusters), where K is the number of clusters. The algorithm works iteratively to assign each observation to one of the clusters based on the distance between the observations. The outputs of the K-means clustering algorithm include (a) the centroids of the K clusters, which can be used to label new data, and (b) individual data points assigned to a single cluster.

The algorithm for K-means Clustering (Han et al., 2012, pp. 451–454) requires a starting position for the K cluster centres. The function *K-means* either randomly selects K tuples from all tuples to be seed values or alternatively accepts user-specified starting seed values. Each remaining tuple is assigned to the cluster that is most similar based on a distance measure such as the Euclidean distance.

For each cluster, the algorithm then iteratively computes a new mean using the tuples assigned to the cluster in the previous iteration. All tuples are reassigned using the updated mean as the new cluster centre. The algorithm continues to iterate until the assignment of tuples to clusters is unchanged from the previous iteration (Han et al., 2012, p. 377).

Classification of Event Outcomes

The two different methods described above are implemented as four different response variable "modes" within this research. Each response mode is evaluated independently resulting in separate analysis of variance (ANOVA) experiments, each with a set of comparable results for the exploratory data set.

Response variable mode I: Binary – bin method

All events are arranged in order by magnitude of return. Any event with an above average return or a positive return (> 0%), whichever is greater, is labelled as *favourable*, and all events that have a less than average return or negative return (<= 0%) are considered *unfavourable* (see *Figure 9*). The number of events contained in a profile is dependent on the unique financial security and trading system combination. The average return refers to the average of the events contained within the specific profile.

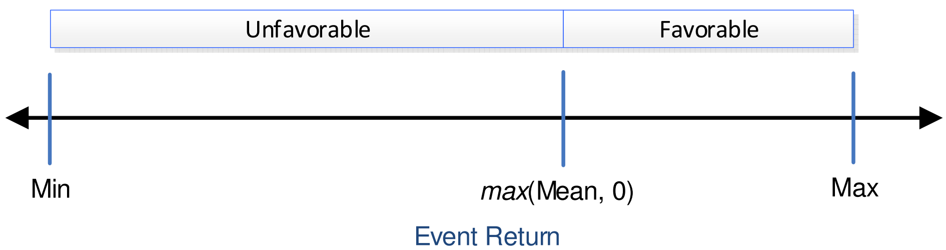


Figure 9. Binary classification, binning method.

Response variable mode II: Multi-bin method

All events are arranged in order by magnitude of return. All events with an above average return or a positive return (> 0%), whichever is greater, are considered at least favourable, and all events that have a less than average return or negative return (<= 0%) are considered at best unfavourable. The top portions of the favourable events are a subset considered *excellent*, and the bottom portion of the unfavourable events are a subset considered *terrible*. Each subset includes at least 10% of the events (*Figure 10*).

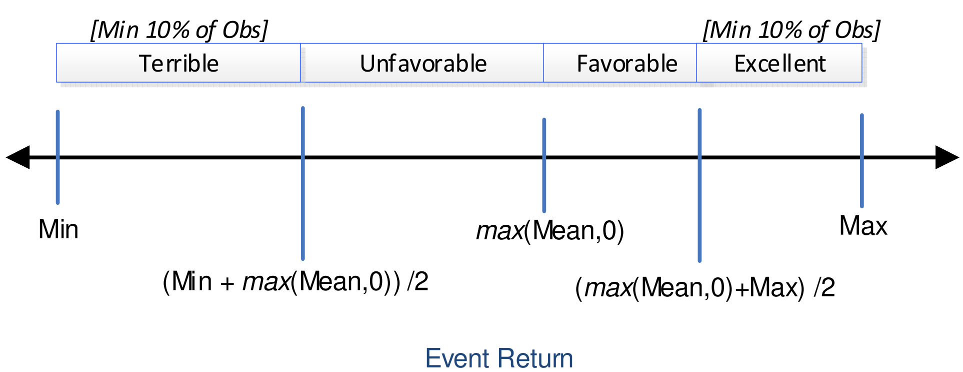


Figure 10. Multi-classification, binning method.

Response variable mode III: Binary – cluster method

Rather than predetermining bin and event outcome classifications, K-means clustering allows groups to be formed. The clustering response mode considers both the return and the duration of the event, where duration is the number of periods the event is in an active state. The initial centres (seeds) of the K-means clustering algorithm are defined using the logic illustrated below with K = 2. Unlike binning, there is no guarantee the events will be grouped into definitively favourable or unfavourable outcomes *Figure 11* shows the positions of the initiating seeds for the k-Means clustering algorithm with K=2.

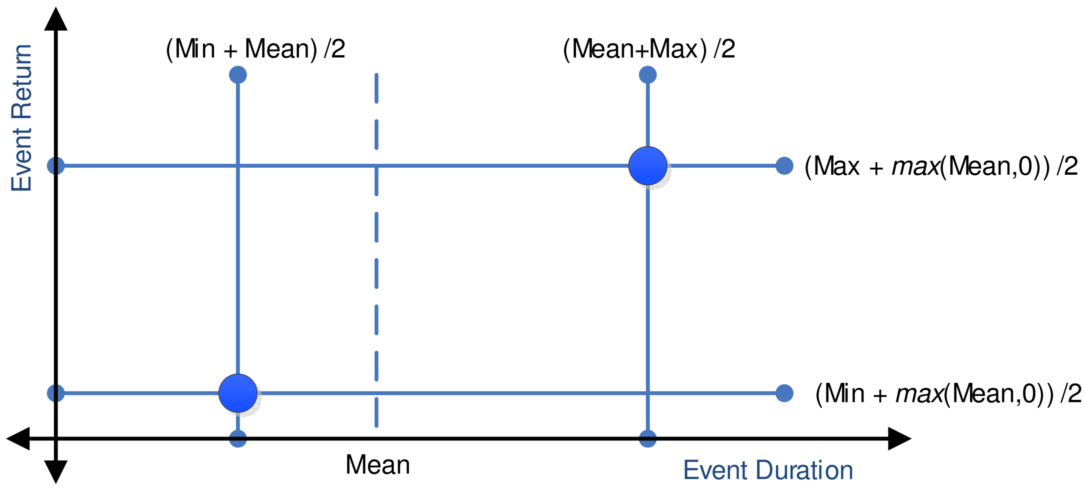


Figure 11. Binary classification, cluster method.

Response variable mode IV: Multi – cluster method

Similar to binary clustering, the seeding centres of the multi-class clustering response mode are set based on the grid coordinates, as shown in *Figure 12*. The clusters evolve over the K-means process and there is no minimum percentage of events per group. The six resulting clusters are not known in advance and may vary significantly from the seeding centre coordinates shown in *Figure 12*.

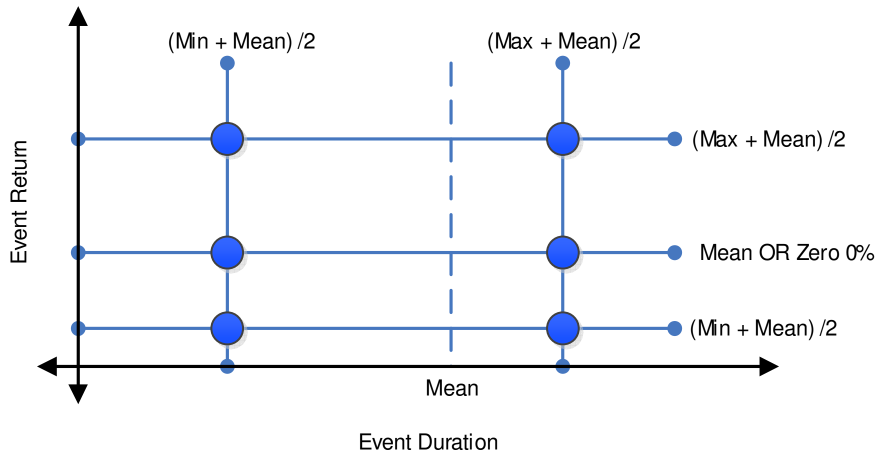


Figure 12. Multi-classification, cluster method.

Figure 13: Formed Clusters Using All Events shows the actual formed clusters of the verification set created by this research[1]. Clusters were assigned ordinal labels based on average event return within the clusters. The clusters zeta, eta, iota, kappa, ksi, and psi are ordered from least average event return to greatest average event return, respectively.

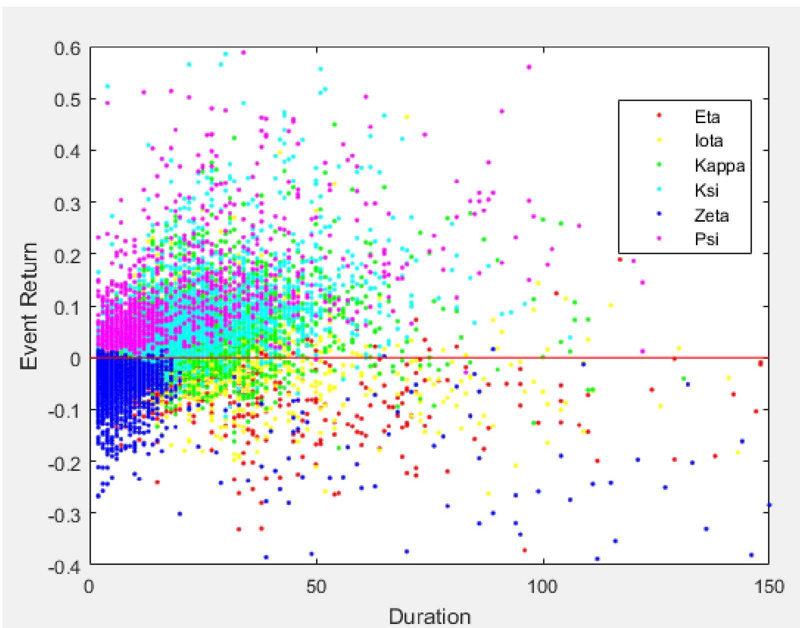


Figure 13: Formed Clusters Using All Events

[1] across all profiles.

For readability, the clusters were relabelled to be consistent with the binning approach, replacing the Greek labelling found within the *application* implemented in support of this research. Table 3 provides the relation between the research labels using this thesis and the Greek letters used in the research application.

Table 3

Response Variable: Cluster Labelling

Research Label	Greeks Label
Terrible	zeta
Undesirable	eta
Unfavourable	iota
Favourable	kappa
Desirable	ksi
Excellent	psi

Map of Related Theoretical Concepts

Figure 14, provided below, illustrates the relationship of the data science concepts utilized within this research. This research applies the theoretical concepts defined above for market events based on technical trading systems in combination with financial securities to develop profiles. The intent of the research is to compare performance metrics of developed classification models through statistical analysis to assess if ensemble machine learning models applied as a prediction performs better than random chance.

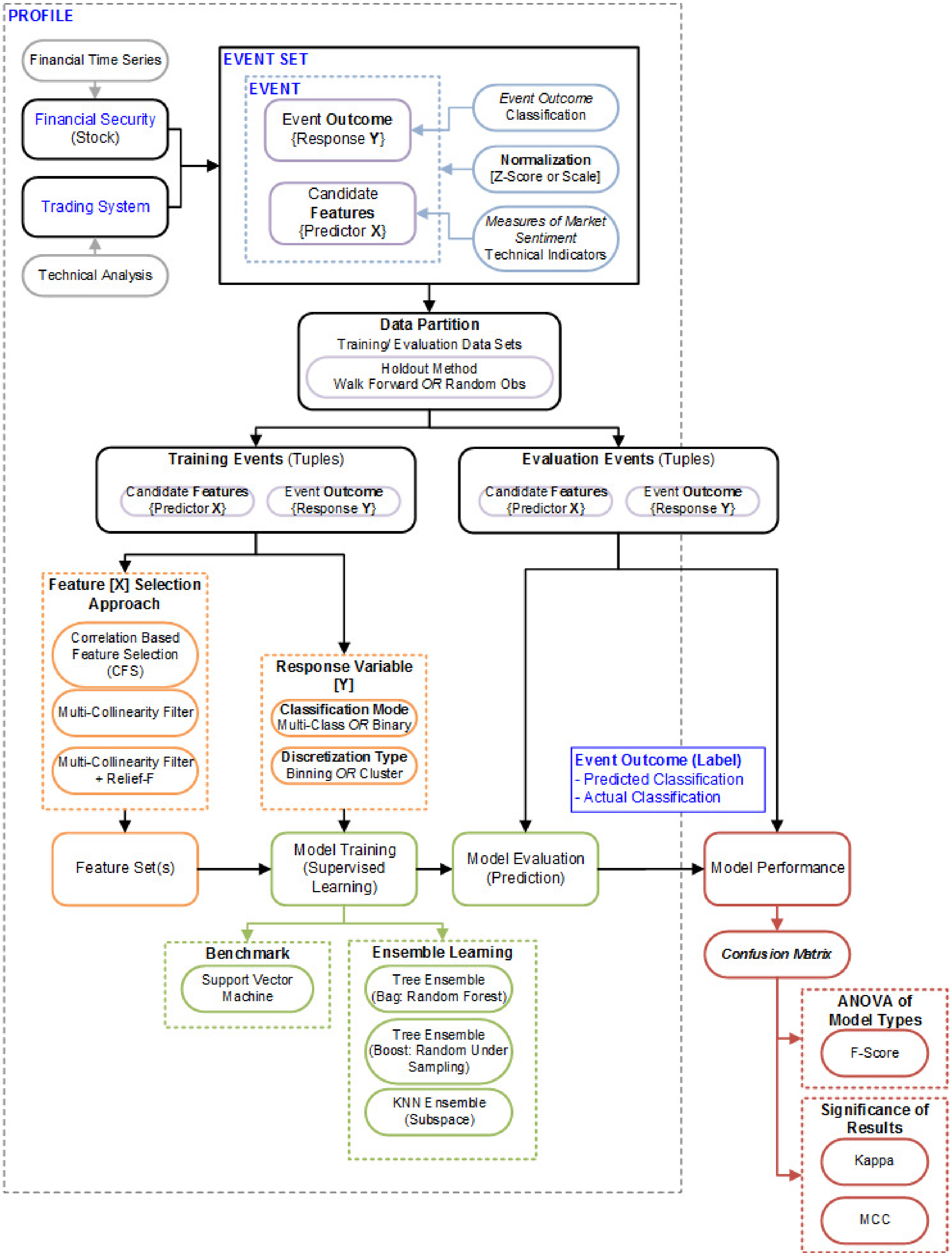


Figure 14. Map of related theoretical concepts.

CHAPTER IV: RESEARCH METHODOLOGY AND SYSTEM DESIGN

Purpose of Study

This research investigated the prospect of classifying the direction of change in price equilibrium using technical indicators as measures of market sentiment. The performance of each classification model was evaluated using the same set of profile events. All events are treated as discrete and unrelated, with the candidate features accounting for the time between events as well as the autocorrelation and heteroskedastic properties of the time series data. The researcher hypotheses the number of events, as well as the demographics of the events, likely vary depending on the financial security and technical trading system. Therefore, the financial security and the technical trading system were identified as probable factors that could influence the behaviour of the classification models.

Factor A: Financial security

To examine this factor, the researcher explored the following question: Does the performance of the classifier models vary with financial security? In doing so, the follow assumptions were made:

- Characteristics of the feature set and event outcome demographics will likely vary with the underlying financial security.
- Likely interaction between financial security and trading system.

The S&P Composite 1500 acted as a proxy population for the market as a whole. It is possible that financial security pricing associated with different market capitalization may have different characteristics. As a result, this research employed an informal stratified random sampling approach by selecting securities at random from each of the composing market indexes of the S&P Composite 1500, namely the S&P SmallCap 600, S&P MidCap 400, and S&P LargeCap 500.

To be randomly selected, securities must have been listed on the exchange for at least 20 years to ensure there is sufficient data for the purposes of this research experiment. If a randomly chosen security did not have 20 years of data, the selected security was discarded and a replacement was randomly drawn. There were no restrictions or special considerations for the demographics of security sector or industry representation.

Factor B: Trading system

To examine this factor, the researcher explored the following question: Does the performance of the classifier models vary with the selected technical trading system? In doing so, the following assumptions were made:

- The technical trading system defines market events differently.
- Demographics of event set will likely vary with different trading systems.

The trading system is an important factor, as it denotes the beginning and end of a market event. Although this research utilized three relatively simple and common trading systems[2], the researcher recognized there are numerous other trading systems and countless combinations of technical indicators that can be combined to create new trading systems.

---

[2] Simple Moving Average, Two Simple Moving Averages, and Bollinger Bands

Factor C: Ensemble classification model

To examine the Ensemble classification model as a factor, the researcher explored the following two questions:

1. Does ensemble classification performance vary with ensemble type and feature selection method? In exploring this question, the following assumptions were made:
    - Ensemble methods are expected to vary in performance.
    - Model performance will likely vary dependent on feature selection method.
2. Do any of the classification models perform more favourably than random chance?

This research investigated if it is possible to classify the outcome of an event using a diverse set of technical market indicators as predictors within a classification model. The primary research question investigated if different ensemble classification models, inclusive of feature selection approach, result in varying performance in classifying event outcomes.

Table 4

Summary of Classification Models Considered

Classifier Model	Summative Description and Highlighted Parameters
Random Forest (RF) (Bag + Random Feature Selection)	Bagging works by training a collection of learners on resampled versions of the data. The resampling is accomplished by selecting a subset of observations *with replacement* for every new learner/member of the ensemble. In addition, every tree in the ensemble can randomly select predictors (features) for decision split criteria, which is a technique referred to as *random forest*. In bagging, ensemble learners are developed independently.
Boost (Random Under Sampling) Classification Tree Ensemble	Boosting implements forward stepwise additive modelling. The boosting class of algorithms starts with an empty ensemble and incorporates new learners sequentially. In boosting, each new learner is influenced by the performance of the learners previously built. Boosting encourages new learners to become experts for classes handled incorrectly by earlier learners by assigning greater weight to those instances.
K-Nearest Neighbour (KNN) (Subspace)	The predictions of a nearest-neighbour classifier depend on the distances between the target observation and previous observation. The distance is determined based on the selected features. Nearest-neighbour ensemble classifiers randomly select subsets of attributes. This approach is called the random subspaces method.
Support Vector Machine (SVM) [Benchmark comparison]	The purpose of an SVM benchmark is to determine if an improvement in classification performance occurs as a result of an ensemble of models relative to an alternative, non-ensemble classification approach.

Randomized factorial design (ANOVA)

To evaluate the difference in performance among the considered classification models, this research employed a three-factor factorial experiment. The experiment considered three different ensemble classification models and a SVM classification model used as a comparison benchmark. There were six technical trading systems applied to 30 randomly selected securities in a data set used to explore the formation of the response variable, and a separate 90 randomly selected securities to address the research questions explored in this research work.

There were three primary factors identified, namely, (a) the randomly selected securities, (b) the technical trading systems used to identify market events, and (c) the classification models used to predict the outcome of an event. The definition of a classification model in this research is inclusive of the feature selection method, for example random forest – relief-F, where random forest is the ensemble method and relief-F is the feature selection approach (see Table 5).

Table 5

Summary of ANOVA Components

Factor A: Security Explore (3 x 10 = 30) Verify (3 x 30 = 90)	Factor B: Trading System 10 and 20-day periods (3 x 2 = 6)	Factor C: Classification Models (4 x 4 = 16)	Feature Selection Approach
- S&P SmallCap 600 - S&P MidCap 400 - S&P 500	- Simple Moving Average - Two Simple Moving Averages - Bollinger Bands	- Ensemble: Tree/Bag (RF) - Ensemble: Tree/Boost - Ensemble: KNN/ Subspace - Support Vector Machines	- Correlation-Based Feature Selection (CFS) - Multicollinearity Removed + Relief-F - Multicollinearity Removed - All Features (no feature selection)
		Each model type has x 4 permutations, one for each Feature Selection Method	

Note. CFS = Correlation-Based Feature; KNN = K-Nearest Neighbour; RF = Random Forest.

The approach taken to model events ought to be applicable to any trading system and security combination. Ideally, it is desirable for the performance of the ensemble models (inclusive of the feature selection method) to be independent of the selected securities and trading systems. Although this is desirable, it is very likely the security and trading system will be significant factors in the performance of the ensemble classification models.

Response variable types and modes

This research investigated four different response modes using a smaller exploratory data set that consisted of 30 securities. Based on findings observed in the exploratory data set, the response mode "Multi – Cluster," which produced the most favourable results, was then further explored using a separately selected 90 securities in the verification data set, consisting of 30 random financial securities per subgroup of the S&P.

Exploratory data set

The exploratory data set is used to examine the different types of response modes. Table 6 presents the response types and modes by scenario.

Table 6

Response Type and Mode Scenario Summary

Scenario	Response Type	Response Mode
1	Binning	Multi-Class
2	Binning	Binary
3	Clustering	Multi-Class
4	Clustering	Binary

This research examined 2,880 models in total for each exploratory data set

- Factor A: 10 X 3 (S&P SmallCap 600, S&P MidCap 400 and S&P LargeCap 500) = 30 Securities.
- Factor B: 3 Systems (BB, SMA, SMA2) * 2 Time Periods (20 days, 10 days) = 6 Trading Systems.

- Factor C: 16 classification models (4 x 4):
  - Three ensemble classification models (RF, RUS, KNN) and one benchmark model (SVM).
  - Four feature selection approaches (all features, CFS, multicollinearity removed, relief-F).

The 180 profiles (Factors A and B) multiplied by the 16 classification models (Factor C) resulted in 2,880 models in total.

Verification data set

After the exploratory data results were reviewed, the response model “multi-classification – clustering” was selected. The hypotheses of this research were then evaluated using the verification set and the response mode multi-classification – clustering.

The verification data set in this research examined 8,640 models in total using only the one response variable scenario (Multiclassification – Cluster). The verification data set comprised 540 profiles—90 Factor A profiles multiplied by six Factor B profiles:

- Factor A: 30 X 3 (S&P SmallCap 600, S&P MidCap 400 and S&P LargeCap 500) = 90 Securities.
- Factor B: 3 Systems (BB, SMA, SMA2) * 2 Time Periods (20 days, 10 days) = 6 Trading Systems.
- Factor C: 16 classification models (4 x 4).:
  - Three ensemble classification models (RF, RUS, KNN) and one benchmark model (SVM).

- Four feature selection approaches (all features, CFS, multicollinearity removed, relief-F).

The 540 profiles (Factors A and B) multiplied by the 16 classification models (Factor C) resulted in 8,640 models in total. The following *Figure 15* provides an overview of the research components described above.

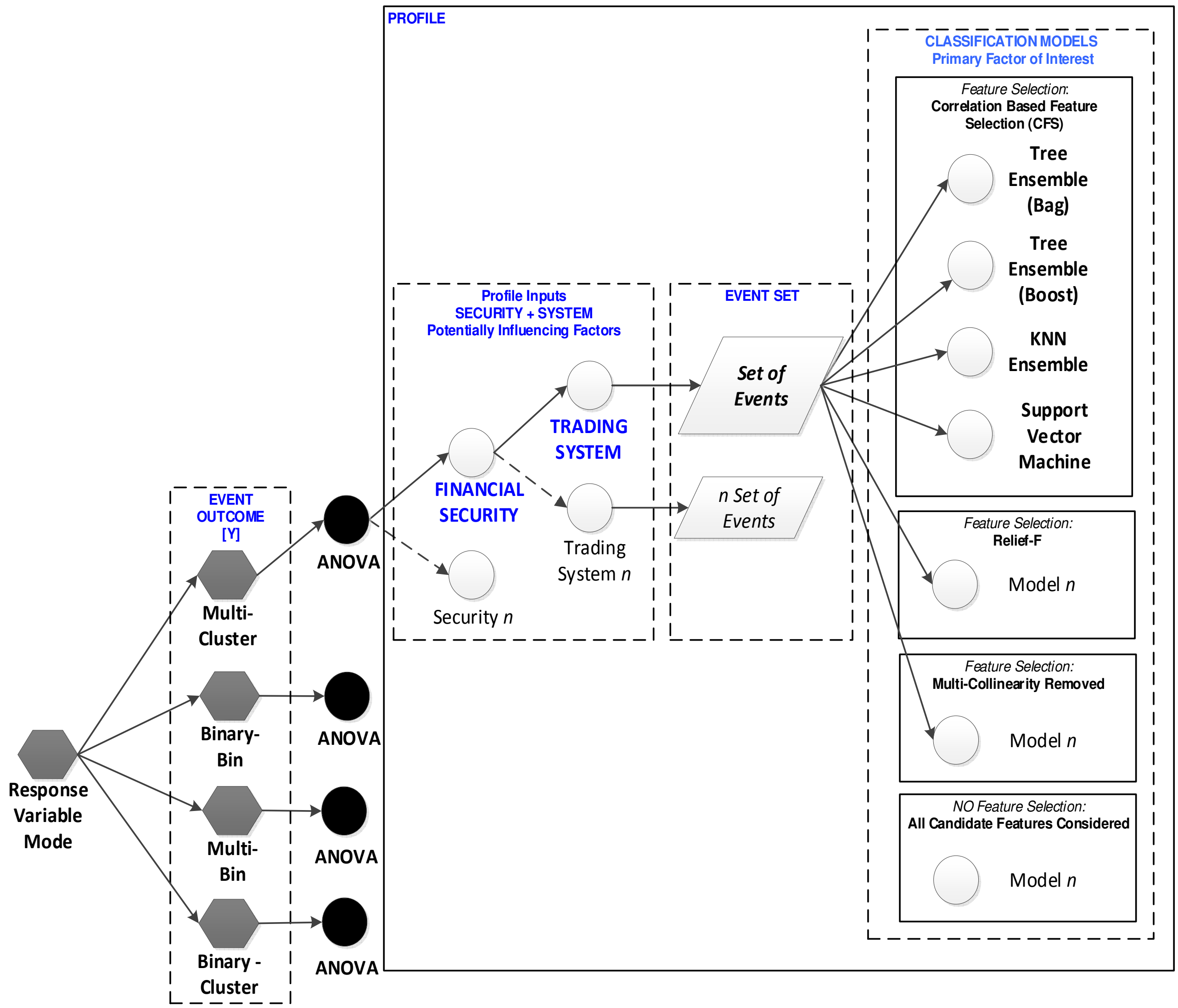


Figure 15. Overview of research components.

For simplicity and process time, the researcher elected to use the two-stage approach of (a) evaluating the response mode using an exploratory data set followed by (b) using a verification set to assess the hypotheses of this research. The response mode was expected to have an impact on the classification model performance. By including the response mode, the ANOVA would have become four dimensional, making the interpretation of any results cumbersome without significant gain.

Furthermore, the processing time of a scenario was strongly influenced by the number of profiles, or more specifically, the time required to generate the profile event set (and corresponding candidate feature observations for each event), rather than the time required to generate the models within the profile. Using the exploratory data set allowed the identification of the most favourable response mode based on high level averages, while the verification data set allowed for more profiles and events to be used for detailed analysis and discussion around the hypotheses.

Formal Hypotheses

This research was structured around a question of difference, specifically, do varying classifier models perform differently among the chosen trading systems and randomly selected securities within the selected response mode. For the response variable mode selected:

1. H_{01}: $\mu_{\text{Tree (Bag) - CFS}} = \mu_{\text{Tree (Bag) – Relief-F}} = \mu_{\text{Tree (Bag) – Collinearity Removed}} = \mu_{\text{Tree (Bag) – All Features}} = \mu_{\text{Tree (Boost) - CFS}} = \mu_{\text{Tree (Boost) - Relief-F}} = \mu_{\text{Tree (Boost) - Collinearity Removed}} = \mu_{\text{Tree (Boost) - All Features}} = \mu_{\text{KNN (Subspace) - CFS}} = \mu_{\text{KNN (Subspace) - Relief-F}} = \mu_{\text{KNN (Subspace) - Collinearity Removed}} = \mu_{\text{KNN (Subspace) - All Features}} = \mu_{\text{SVM – CFS}} = \mu_{\text{SVM – Relief-F}} = \mu_{\text{SVM - Collinearity Removed}} = \mu_{\text{SVM - All Features}}$; there is no difference in the mean classifier performance among varying classification models.

   H_{a1}: There is a difference in the mean classifier performance among varying classification models.

2. H_{02}: $\mu_{SecurityA} = \mu_{SecurityB} = \ldots = \mu_{SecurityN}$; there is no difference in the mean classifier performance among the randomly selected securities

   H_{a2}: There is a difference in the mean classifier performance among the randomly selected securities.

3. H_{03}: $\mu_{System1} = \mu_{System2} = \ldots = \mu_{SystemN}$; there is no difference in the mean classifier performance among the different chosen trading systems.

   H_{a3}: There is a difference in the mean classifier performance among the different chosen trading systems.

4. H_{04}: The trading system has no influence on how the classification models perform.

   H_{a4}: There is an interaction between classification models and the trading systems.

5. H_{05}: The security has no influence on how the classification model affects the classification model's performance.

   H_{a5}: There is an interaction between the security and the classification models.

6. H_{06}: The security has no influence on how the trading system affects the performance of the classification model.

   H_{a6}: There is an interaction between the trading system and the classification models.

Research Methods – Application Framework

The term *application* refers to the MathWorks® (2014) developed MATLAB program as a whole and all other technical implementation created to support this research. The application design is presented in logical processing order, starting with an overview of the application use cases and a high-level processing sequence. The key processing logic used to create events, manage model training and evaluation data, perform feature selection, and generate the supervised machine learning models considered in this research are presented in *Figure 16* and *Figure 17* below and are summarily detailed in Table 7.

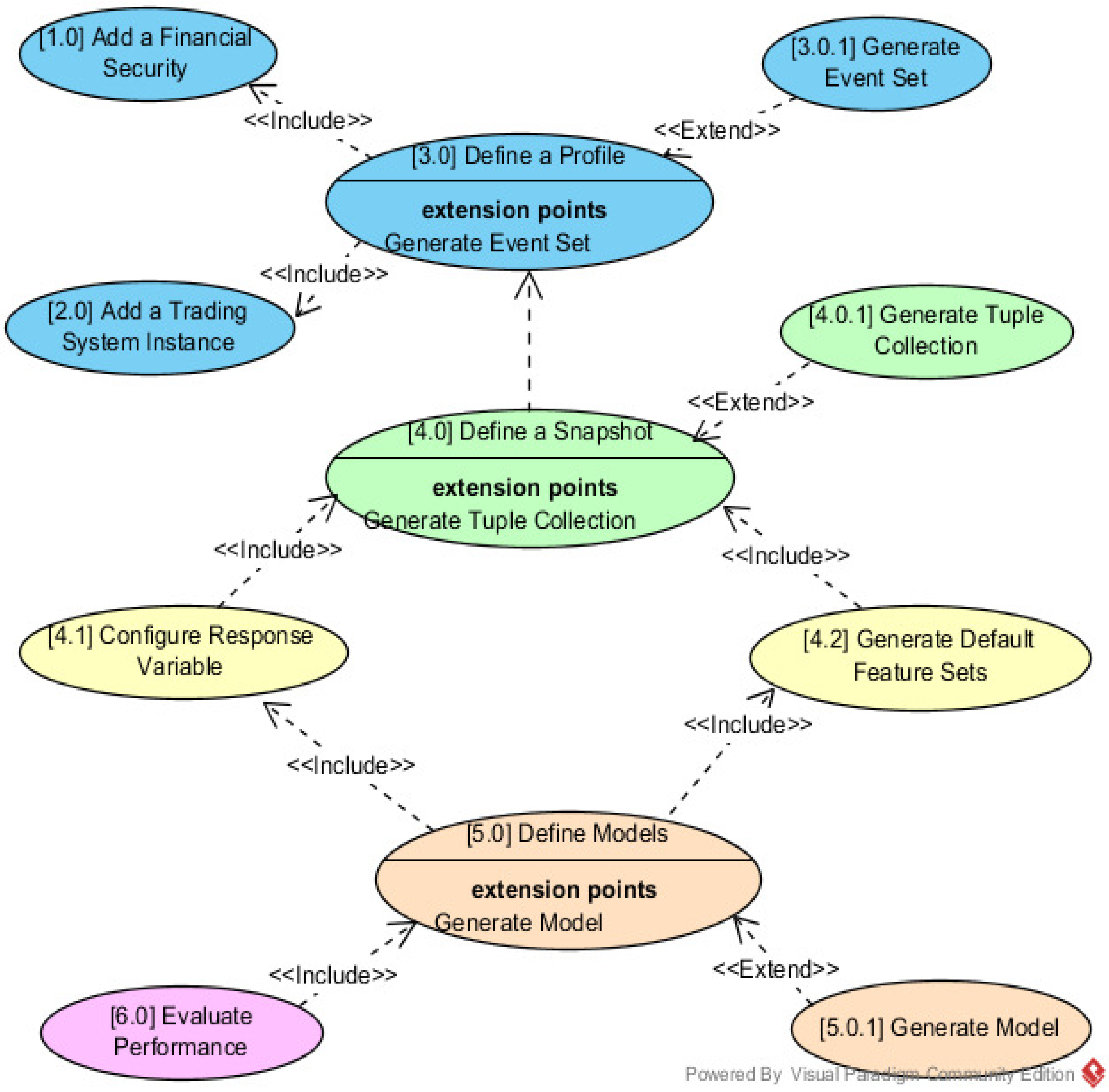


Figure 16. Use cases of the application.

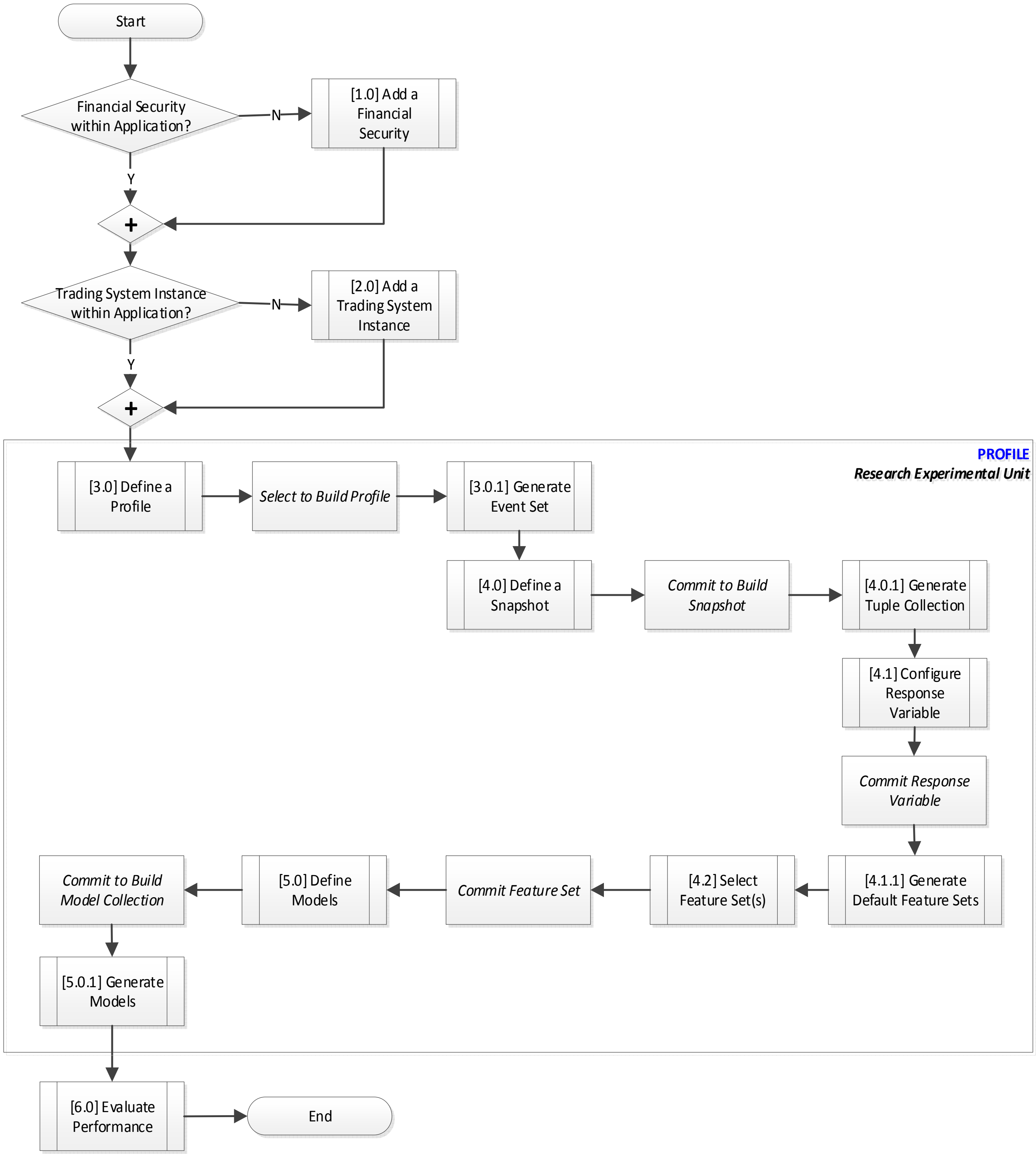


Figure 17. High-level process diagram.

Table 7 *Description of Use Cases and High-Level Processes*

Ref.	Description
1.0	Add a financial security. A financial security is added to the application via an identifying symbol (market ticker). Within this research, only equities on the S&P Composite 1500 were considered for random selection.
2.0	Add a trading system instance. A technical trading system is used to define the beginning and end of an event. A trading system contains an entry trigger and an exit trigger. A trading system instance is created from one of the available trading systems contained in the application by specifying the parameters (e.g., time dimension, etc.) that in turn define the characteristics of the entry trigger and an exit trigger.
3.0	Define a profile. Inputs to a profile include a financial security, a trading system, either a long or short trading strategy, and a predetermined response variable mode. Based on those inputs, a profile contains a collection of events, a snapshot (model data), and a collection of models.
3.0.1	Generate event set. The financial security fetches a time series of historic market quotes from a market data source. The profile uses the entry and exit triggers defined in the trading system to generate the set of profile events. The events are managed in chronological order. Events are consecutively formed with only one current event in a monitoring or active state in a particular profile at any one time; all other events are in an historic state. Each historic event contains entry and exit information as well as a collection of candidate features. *Candidate Features* For each event, at the time the trading system triggers an entry signal and the event transitions to an active state, each candidate feature in all its derivative forms are calculated and recorded.
4.0	Define a snapshot. Historic profile events are used to form the snapshot, which manages the data for the models. Events are added to the snapshot by specifying which events from the profile are to be included in the snapshot. Once all desired events are added into the snapshot, the snapshot will generate a collection of snapshot tuples. A snapshot tuple is a flat record format of event data contained in the event object. It consists of the event outcomes (duration and return) and all candidate features.
4.0.1	Generate tuple collection. The process of creating a snapshot includes transitioning a collection of event objects into a tuple/ flat record format as well as normalizing the candidate features and event outcome data. Candidate features and event returns are

Ref.	Description
	continuous variables. In the classification models, the scale/units of a variable can have an impact on their perceived importance in the models. Normalization provides a common relative scale.
4.1	Configure response variable. Four different response 'modes' were considered in this research: • Multi classification– Bin • Binary classification – Bin • Multi classification – Cluster • Binary classification – Cluster Committing the response variable triggers the generations of the default feature set (or sets). One response variable mode is chosen per application run.
4.2	Generate [default] feature set (or sets). A feature set is a subset of features selected from the pool of available candidate features. Three different filtering approaches are considered within this research, leading to the creation of three distinct feature sets, plus a set containing all features (no feature selection). The feature selection methods considered are the following: • All Features (no feature selection) • Multicollinearity Removed • Multicollinearity Removed + Relief-F • Correlation Feature Selection (CFS) The resulting feature sets contain a collection of chosen candidate features used in the ensemble models to predict the outcome of events.

5.0	Define models.
	A model is a supervised machine learning algorithm that maps the selected features/predictors to an event outcome/class label. A model accepts a feature set containing the historic events' outcomes and corresponding candidate feature data in the form of a collection of snapshot tuples. There are three different ensemble model types considered in this research as well as one non-ensemble comparison:
	• Ensemble model types:
	  o Bagging; Random Forest
	  o Boosting; Random Under Sampling
	  o Subspace; K-Nearest Neighbour
	• Non-Ensemble model type:
	  o Support Vector Machines
	Although the parameters of each ensemble type and the supervised learner contained are each definable/tunable, the parameters of each classification model type have been selected based on the literature reviewed and held

Ref.	Description
	constant among the various generated profiles.
5.0.1	Generate models. Using the committed response variable and the committed features set (sets), the application generates each of the 16 different models. The collection of event tuples within the snapshot are partitioned into two segments, namely training data and evaluation data. The training data are used to create the classification models. The evaluation data are used to measure the performance of the models and evaluate the hypotheses.
6.0	Evaluate performance. Each model generated by the application can be loaded into a reporting module. The reporting module is able to generate a confusion matrix and several classifier performance metrics.
7.0	Research experiment processing. The processing time to run a large number of scenarios is considerable. Correspondingly, a script that automates and parallelized the creation of profiles and the corresponding event sets, selection of feature sets based on the predefined features selection approaches, generate the ensemble models within the profiles, predict event outcomes using the evaluation data partition, and save all data to the central repository was created.

Sample

This research used actual historical market data sourced from a free market data Application Program Interface (API) provided by Quandl (n.d.). Market events for each security was generated using 15 years of historic time series data (2002–2016, inclusive). The number of events and demographics of event attributes were dependent on both the financial security and the trading system. Security pricing was not adjusted for dividends.

Instruments

The following software programs were used for this research:

- MATLAB (MathWorks®, version 2015b)—the main application software of research methods.

- MySQL + Workbench (Oracle version 6.3, 2015)—database storage and retrieval program for profiles and results.
- Visual Paradigm (version 13.1 2016)—used for the design of the application in unified modelling language (UML).

Analysis

This section provides an overview of the analysis completed in support of this research, specifically the architecture of the developed application used to create profiles, manage model data, select features, generate models, and the approach used to evaluate resulting model performance. In the next chapter, the results from the application and model outputs are presented and are explored through an ANOVA and other statistical analysis to address the hypothesis questions associated with this research.

Define a profile [3.0]

A profile is created by applying a technical trading system to a security to generate a collection of historic market events specific to a trading system and a financial security combination. The resultant events are used to create a snapshot of data that is processed through four different machine learning classification models using four different feature sets. The results of the machine learning models were captured within a specific profile and compared to other created profiles. *Figure 18* illustrates the relationships of the components of a profile.

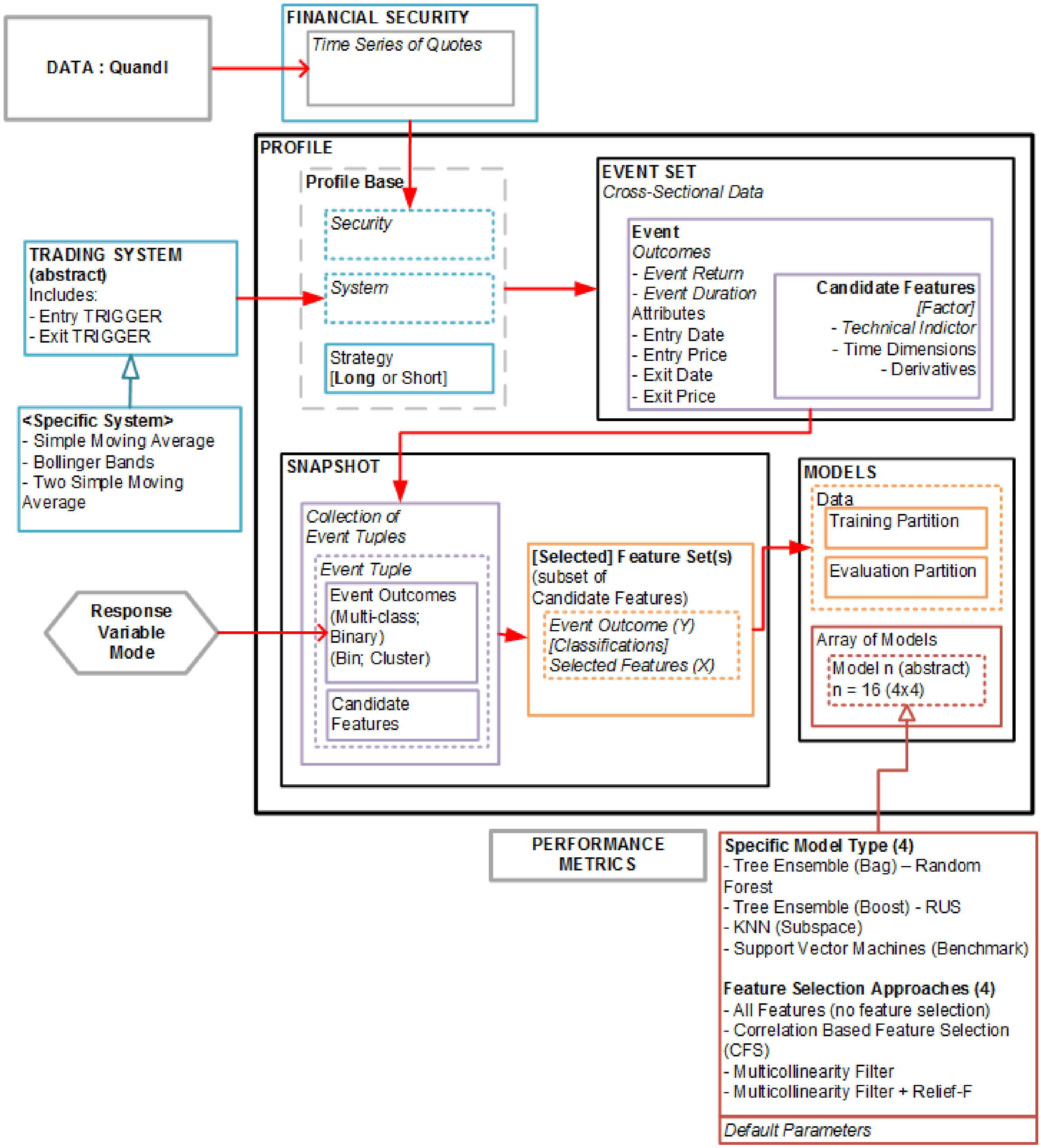


Figure 18. Application object relations.

Generate event set [3.0.1]

An event set was composed of individual events defined by the selected trading system for the financial security of the profile. Based on end-of-day (EOD) data for a specific financial security, the application examined the data to:

1. adjust for any financial security splits within the period.
2. identify the state of the current event (monitoring or active).
3. determine if a state change was triggered (i.e. monitoring to active, or active to historic).

Based on the results of the examination, the application recorded the applicable information and updated the state of the event and/or the associated duration count. *Figure 19* illustrates the chronology of how events were processed and generated.

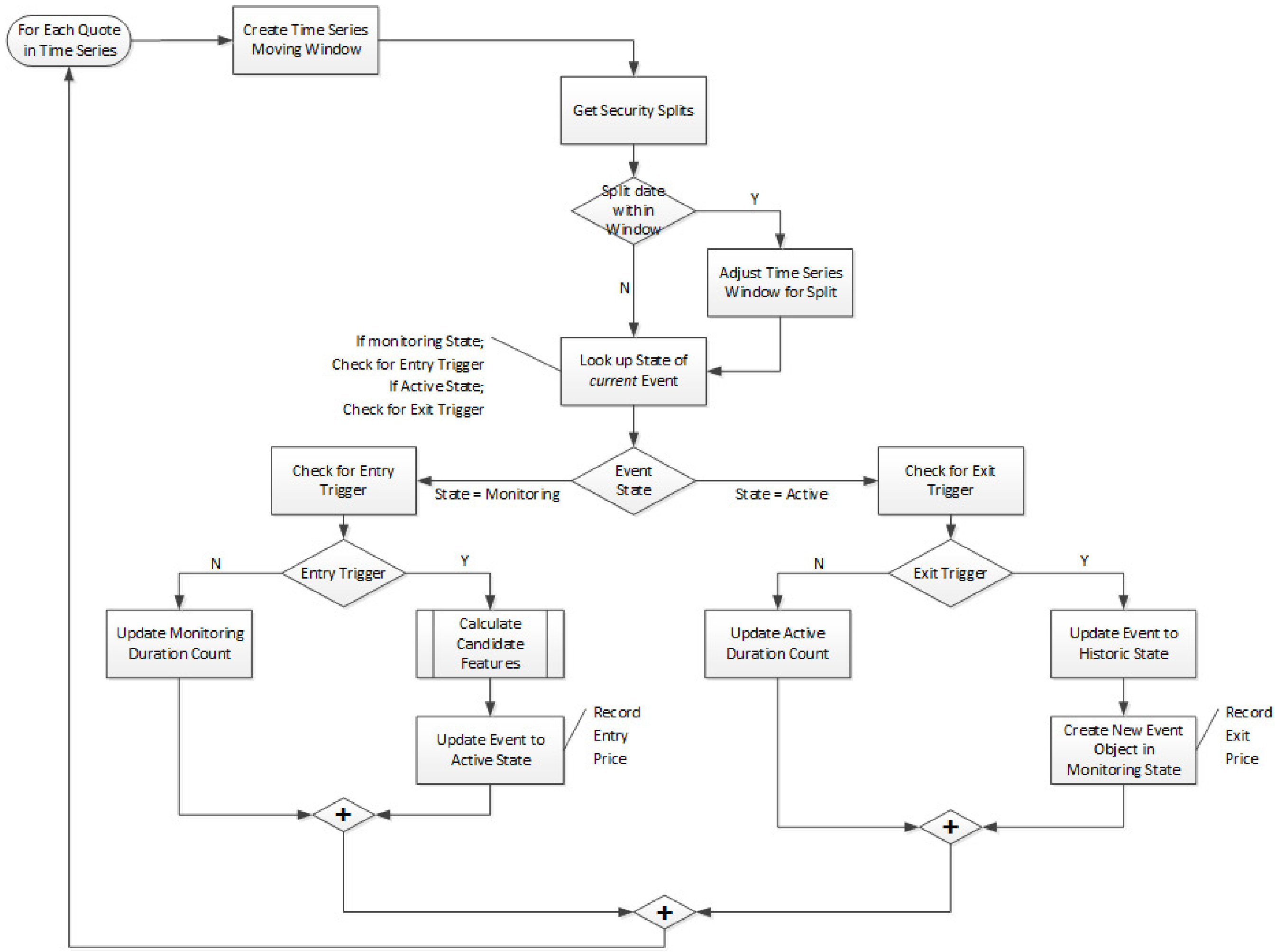


Figure 19. Chronology of event generation.

For each event, at the time the trading system triggered an entry signal and the event transitioned to an active state, each candidate feature in all its derivative forms were calculated and recorded. Each historic event contained entry and exit information as well as a collection of candidate feature values.

Management of splits

A stock that splits has a significant impact on the security price and volume relative to its recent pricing history. Therefore, an event needed to be managed such that it could start and finish under the same pricing pretences without artificial signals resulting from a split. *Figure 20* and *Figure 21* illustrate how stock splits were managed within the application, Table 8 summarizes the illustrations.

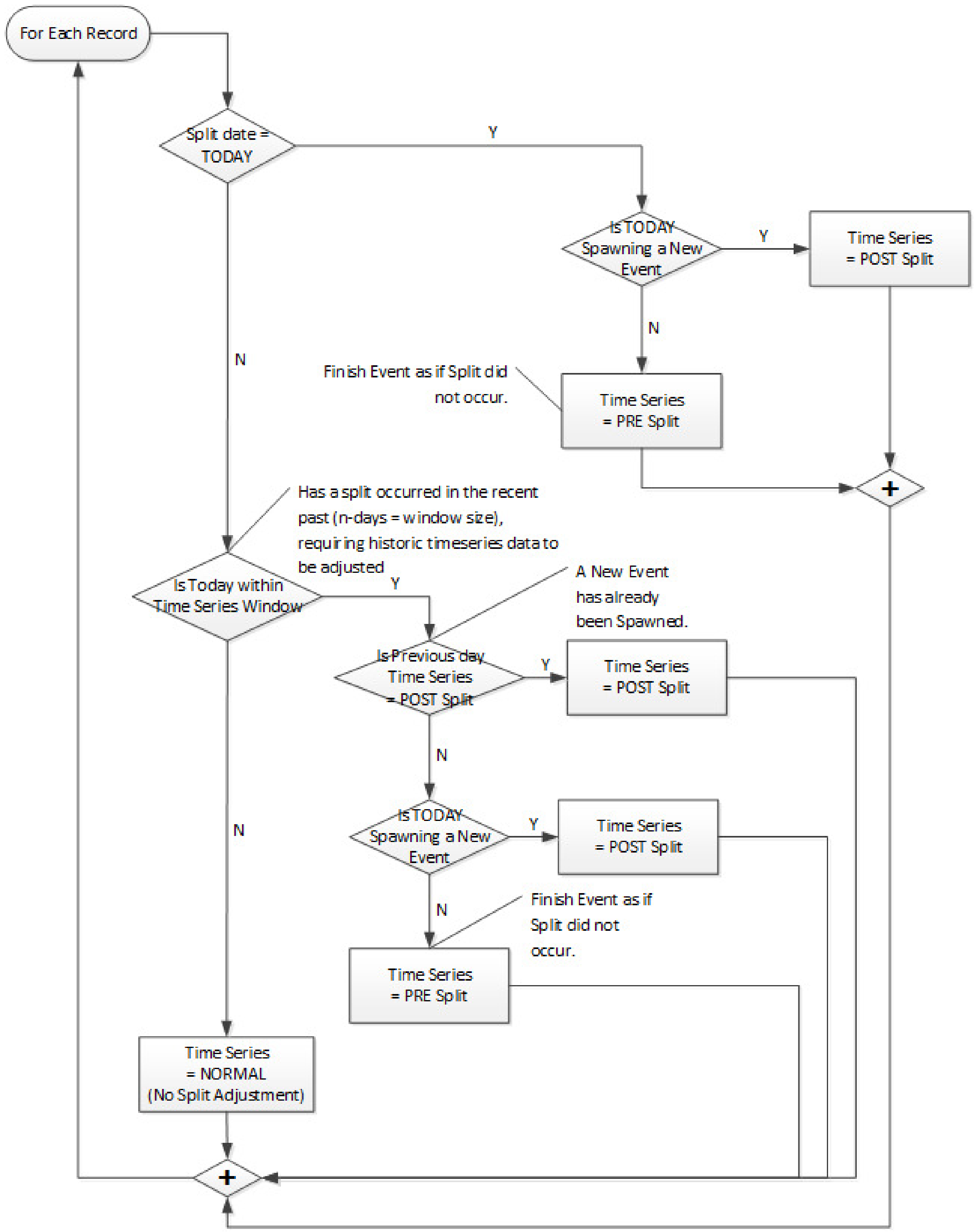


Figure 20. Data management with consideration for splits (process diagram).

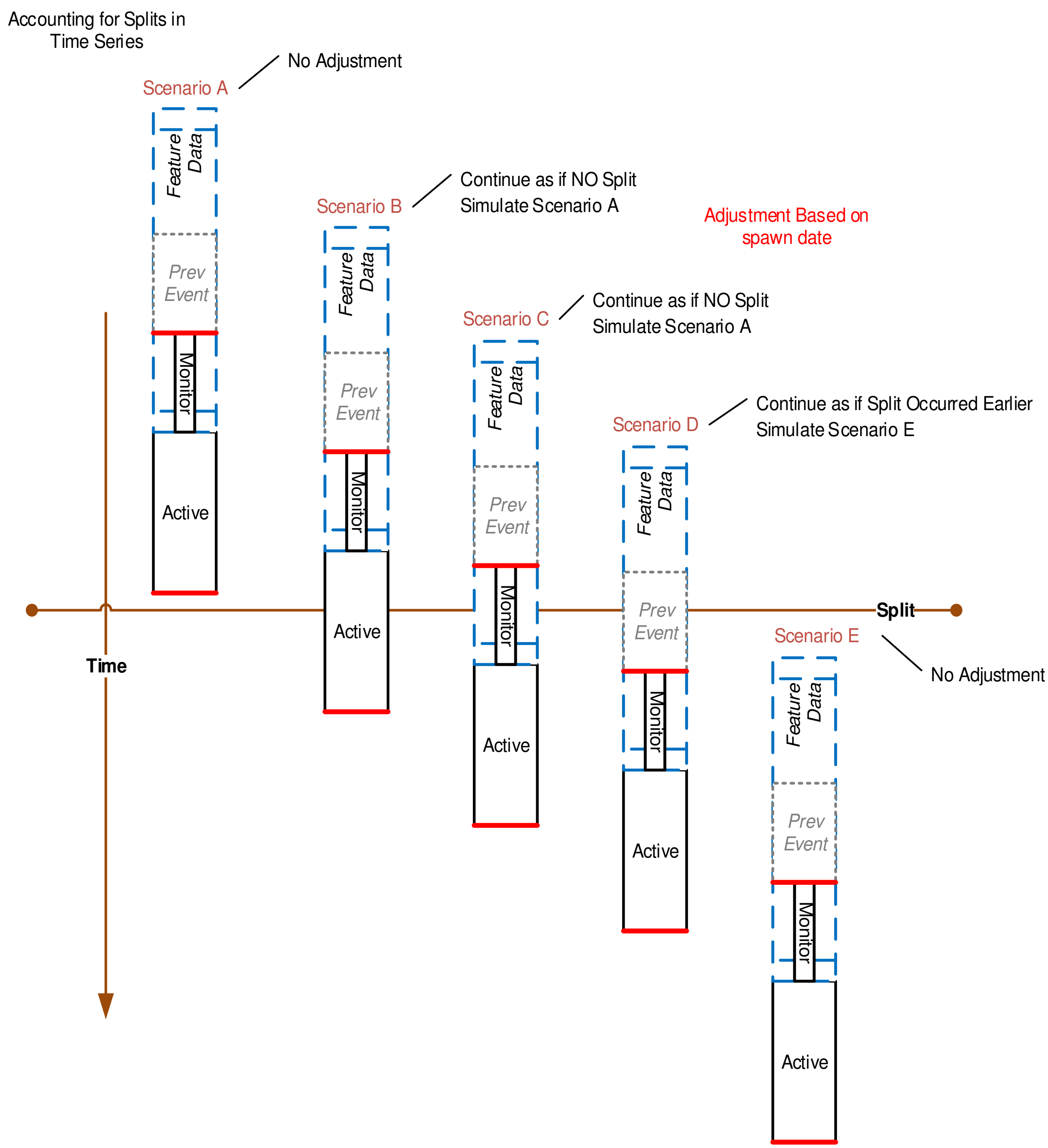


Figure 21. Management of stock splits within the application.

Table 8

Management of Stock Splits in the Application

Scenario	Description
A	Standard scenario—no split has occurred.
B	A stock split occurred while an event is in an *active state* and the trading system is currently monitoring for an exit signal. The financial time series data are adjusted to *reverse the split* allowing the event to complete as if a stock split had not occurred. Security pricing data after the split date are reversed by multiplying by the inverse split ratio.
C	A stock split occurred while an event is in a *monitoring state* and the trading system is currently monitoring for an entry signal. As the event has already been spawn, the event will continue as if the stock split has not yet occurred. All security pricing data after the split date are *reversed* by multiplying by the inverse split ratio.
D	The calculation of candidate features using historic time series data are influenced by the stock split. In this scenario, the historic security pricing is adjusted to simulate the stock split occurring *earlier* such that all candidate features are calculated on a pricing post stock split basis. Security pricing prior to the split date is adjusted by multiplying by the split ratio.
E	Standard scenario—previous split no longer impacts the current event, nor the calculation of the candidate features.

Define a snapshot [4.0]

A snapshot included a collection of events from a profile that became the input data for the supervised learning classification models. For purposes of this research, a snapshot contained all events from a single trading system and financial security pair that occurred within the dates of the period of study. The snapshot formed the data set for the models. The event records contained in the snapshot are referred to as a tuple, which were partitioned into training and evaluation data.

The snapshot is created through the discretization of the event set based on different response variable modes, which were then filtered through the following four different feature selection processes:

- all features (no feature selection);
- multicollinearity removed;
- multicollinearity removed + relief-F; and
- correlation-based feature selection (CFS).

The snapshot presented a common data source for use in the 16 different machine learning models (four model types across four feature selection approaches) for a given period of time with a start date and an end date. The snapshot assumed a set response variable mode for the application, with each trading system/security pair processed through the application. *Figure 22* illustrates the process of creating a snapshot, which managed data and feature selection processes of the models.

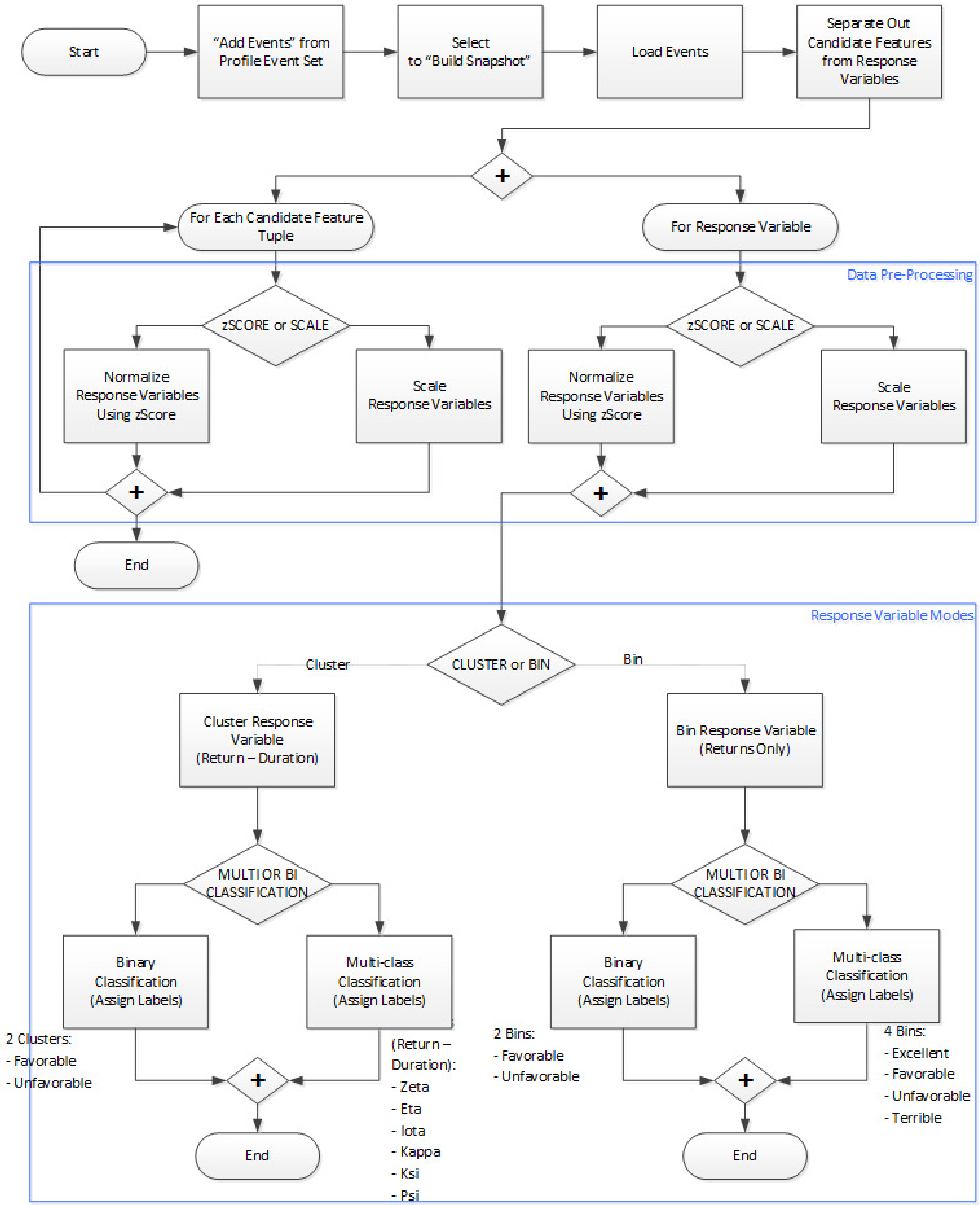


Figure 22. Define a snapshot.

Generate tuple collection (process diagram) [4.0.1]

The event objects added to the snapshot were flattened into records referred to as a snapshot tuple. The snapshot tuples were then managed as a collection of records. Individual tuples within the snapshot were normalized through either standardization (Z-score) or rescaling, which are defined in the subsections that follow.

Data normalization (pre-processing)

Some machine learning algorithms will not work properly without normalization. For example, Breiman (2001), in reference to bagging ensembles, suggested that if the features in a data set do not have a reasonably common standard of measurement, they need to be normalized by subtracting means and dividing by standard deviations, where the means and standard deviations are determined from the training set. It can be argued that this is a relevant step for all models considered in this research project.

The majority of classifiers calculate the distance between two points using a distance measure such as Euclidean. If one of the features has a broad range of values (such as trading volume vs. price), the distance between events will be strongly influenced by this particular feature. To eliminate the units associated with the different technical indicators (candidate features) and better support the assumptions of the models, the event data were normalized so that each feature that was selected contributed fairly to the results. Two methods were considered for data normalization, namely standardization and rescaling.

Standardization (Z-score)

The Z-score is a standardized value that can be interpreted as the number of standard deviations that observation x_i is from the mean. A Z-score value greater than zero occurs when an observation has a value greater than the mean, and a Z-score value less than zero occurs when an observation has a value less than the mean (Anderson, Sweeney, & Williams, 2005, pp. 94–96). The Z-score can be defined as

$$Z_i = \frac{x_i - \bar{X}}{s}$$

where Z_i = z-score for x_i, $\bar{X}$ = mean, and s = standard deviation.

The Z-score for any observation can be interpreted as a measure of the relative position within the data set. The Z-score is used to determine if a particular observation is close to the centre of the data set or far out in one of the tails. This allows for observations in two different data sets with the same z-score to have the same relative location in terms of being the same number of standard deviations from the mean (Anderson et al., 2005, pp. 94–96).

Rescaling

As an alternative to using the Z-score, the observations can be scaled. The simplest method is rescaling the range of feature observations between zero and one [0, 100%]. The general formula is

$$X_i{}' = \frac{x_i - \min(X)}{\max(x) - \min(X)} * 100$$

where x_i is an original value, and x_i' is the normalized value.

Implementation

Both standardization and rescaling were considered for use in the application. The standardization method was selected over the rescaling method for this research and used consistently in all scenarios.

Generate default feature sets [4.2]

Feature selection overview

The application generated thousands of candidate features at the time of each entry signal. Using learning algorithms with all available features can result in deteriorating performance (Aly & Atiya, 2006). Feature selection is the task of selecting a subset of the most appropriate features (model predictor variables) from all available candidate features to reduce the dimensionality of data while still being able to describe the target concept/response variable (MathWorks®, 2014; Robnik-Šikonja & Kononenko, 2003).

There are two paradigms to addressing high dimensionality namely (Wu et al., 2008): feature selection and dimension reduction/consolidation (e.g., principal component analysis - PCA). This research implemented feature selection as opposed to feature reduction, which is the process of selecting a subset of features from all available candidate features. As the candidate features were all derived from the security's price and volume data, there were several highly correlated features. Feature selection was primarily focused on removing non-informative or redundant features from the model (Kuhn & Johnson, 2013, p. 487) based on the following assumptions:

- A relevant feature is both important and non-redundant to the model or concept.
- An irrelevant feature does not affect the model in any way, and a redundant feature does not add any new information (Dash & Liu, 1997).

Feature selection directly chooses a subset of the features, thereby preserving their original meaning. Feature selection is preferable to feature transformation when the original units are important to the meaning of the features and the modelling goal is to identify an influential subset (MathWorks®, 2014). As technical indicators are transformations of financial time series data, it did not make sense to employ a dimension reduction technique such as PCA that would again transform the data and lose the meaning of technical indicators.

Given the number of candidate features, the use of ensembles, and multiple models under evaluation, this research employed a filtering method (as opposed to wrapper methods) to determine the feature set. Each of the ensemble models evaluated was provided the same feature set to allow comparisons to occur.

Feature selection methods

Feature selection is generally divided into two approaches, namely filter methods and wrapper methods. Filter methods select a subset of features as a pre-processing step independent from the chosen learning machine algorithm (model). In contrast, wrapper methods utilize the learning machine algorithm as a black box to score subsets of features according to the resulting performance of the trained classification model (Dash & Liu, 1997; Guyon & Elisseeff, 2003).

Although wrapper methods generally improve predictor performance compared to simpler filtering methods (e.g., correlation), the improvements are not always significant (Guyon & Elisseeff, 2003). The wrapper approach quickly becomes computationally cumbersome depending on the dimensionality of the data. Domains with large numbers of input variables suffer from the curse of dimensionality and multivariate methods that may over fit the data (Guyon & Elisseeff, 2003; Kuhn & Johnson, 2013, p. 490). Given the number of models evaluated, the use of ensembles, and the number of parameters that require tuning, wrapper methods were difficult to employ in this research.

Feature selection approaches

This research implemented three different feature selection approaches to evaluate the impact of feature selection on model performance. In addition, a feature set with all candidate features was considered.

Commonly, filter methods evaluate each candidate feature independently and consequently risk selecting redundant features (i.e., highly correlated), while at the same time risk omitting important interactions between variables. Using different feature selection methods may lead to different features being selected, and thus affects the prediction model performance. *Figure 23* provides a visual representation of the feature selection methods used in this research.

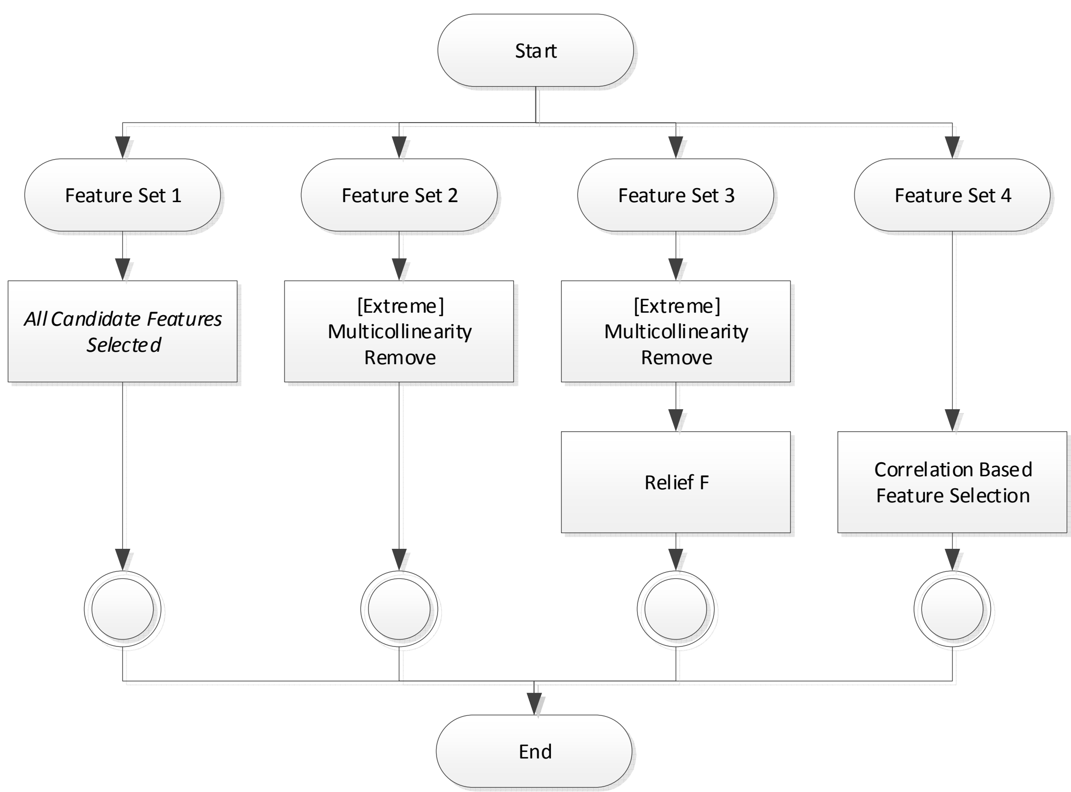


Figure 23. Feature set approaches.

Approach 1: No feature selection (all features)

Each candidate feature generated at the time the event transitions to an active state measures an aspect of market sentiment, each time dimension helps to reflect the nature of the time series data, and each of the derivatives address how the sentiment is changing while directly attempting to account for autocorrelation and heteroskedasticity. Given that each candidate feature has meaning and provides some information, it was, therefore, appropriate to consider "all features" as one of the feature sets investigated.

Approach 2: All features with extreme multicollinearity removed

Collinearity is the technical term for a pair of candidate features that have a substantial correlation with each other. It is also possible for multiple features to be correlated at once (Kuhn & Johnson, 2013, p. 45). Correlation between features is a significant issue in machine learning. Using highly correlated features generally adds little information to the learning process and can have a negative impact on the inductive algorithm (Aly & Atiya, 2006). As a result, it is important to remove predictors that have excessively correlated relationships in prediction models.

A heuristic approach to dealing with multicollinearity is to remove the minimum number of predictors to ensure that all pairwise correlations are below a specified threshold (e.g., 0.8). Kuhn and Johnson (2013) suggested the following algorithm, which was implemented:

1. Calculate the correlation matrix of all considered features.
2. Determine the two features associated with the largest absolute pairwise correlation (Features A and B).

3. Determine the average correlation between Feature A and all other Features. Repeat for Feature B.
4. If Feature A has a larger average correlation among all other features, remove it; otherwise, remove predictor B.
5. Repeat Steps 2–4 until no absolute correlations among the features are above the threshold. (pp. 46–47)

Approach 3: Extreme multicollinearity removed and Relief F

The relief-F algorithm is a heuristic estimator that does not assume that candidate features are independent of each other (Kononenko, Šimec, & Robnik-Šikonja, 1997). Relief-F is aware of the contextual information and can estimate the quality of attributes in domains that have dependencies between attributes (Robnik-Šikonja & Kononenko, 2003).

Relief-F is a feature weight-based algorithm inspired by class-based machine learning algorithms. Similar in concept to KNN, in its simplest form, relief-F randomly picks a sample of tuples, and for each tuple in the sample, it finds the nearest hit and nearest miss based on a Euclidean distance measure (Robnik-Šikonja & Kononenko, 2003). The nearest hit is the tuple having the minimum Euclidean distance among all tuples of the same class. The near miss is the tuple having the minimum Euclidean distance among all tuples of different class (Dash & Liu, 1997). The relief-F algorithm can be configured to search for k near misses for each difference class and average their contribution for updating the estimate (Kononenko et al., 1997).

Relief-F is robust if the number of nearest neighbours (k) remains relatively small in relation to the number observations. If it is too small it may not be robust enough, especially with more complex or noisy concepts (Robnik-Šikonja & Kononenko, 2003). With increases in the number of k nearest hits/misses the correlation of relief-F's estimate with other impurity functions also increases (Kononenko et al., 1997). The appropriate selection of KNNs is dependent to the problem complexity, the amount of noise in the data, and the number of available tuples per class. Relief-F can consider non-uniform cost of misclassification by changing the weights of the attributes to reflect the cost-sensitive prior probabilities (Robnik-Šikonja & Kononenko, 2003). Robnik-Šikonja and Kononenko (2003) suggested that using log *n* nearest neighbours generally gives satisfactory results.

One major limitation of relief-F is that it does not help with redundant features and, hence, generates a non-optimal feature set size in the presence of redundant features (Dash & Liu, 1997). The relief-F algorithm is sensitive to duplicated attributes, as duplicates change the problem space in which the nearest neighbours are searched (Robnik-Šikonja & Kononenko, 2003). With the increasing number of replicate attributes, the quality of estimates will decrease as the replicated attributes affect the distances between tuples (Kononenko et al., 1997).

In this research, the candidate features were first filtered to remove extreme collinearity among the candidate features prior to executing the relief-F algorithm to address concerns relating to redundant features.

It is desirable to have KNNs be small but robust with noisy candidate features. To balance these requirements, a unique approach of applying relief-F was created. As shown in *Figure 24*, the relief-F algorithm was performed multiple times, specifically $\log_2$(minimum class count) times. For example, if there were four possible event outcomes (classes) and the counts of each class were as follows: 32 exceptional, 42 favourable, 70 unfavourable, and 40 terrible. Then k was selected to be Log_2(32 exceptional) = 4. The relief-F algorithm was then run four times. During the first iteration, K was specified to 1, the second iteration K specified to 2, and so on, until K is specified to 4 neighbours. The average feature weight was then determined for each candidate feature and a *t*-test was used to find the features that had an above average weight with at least a 90% confidence ($\alpha = 0.1$).

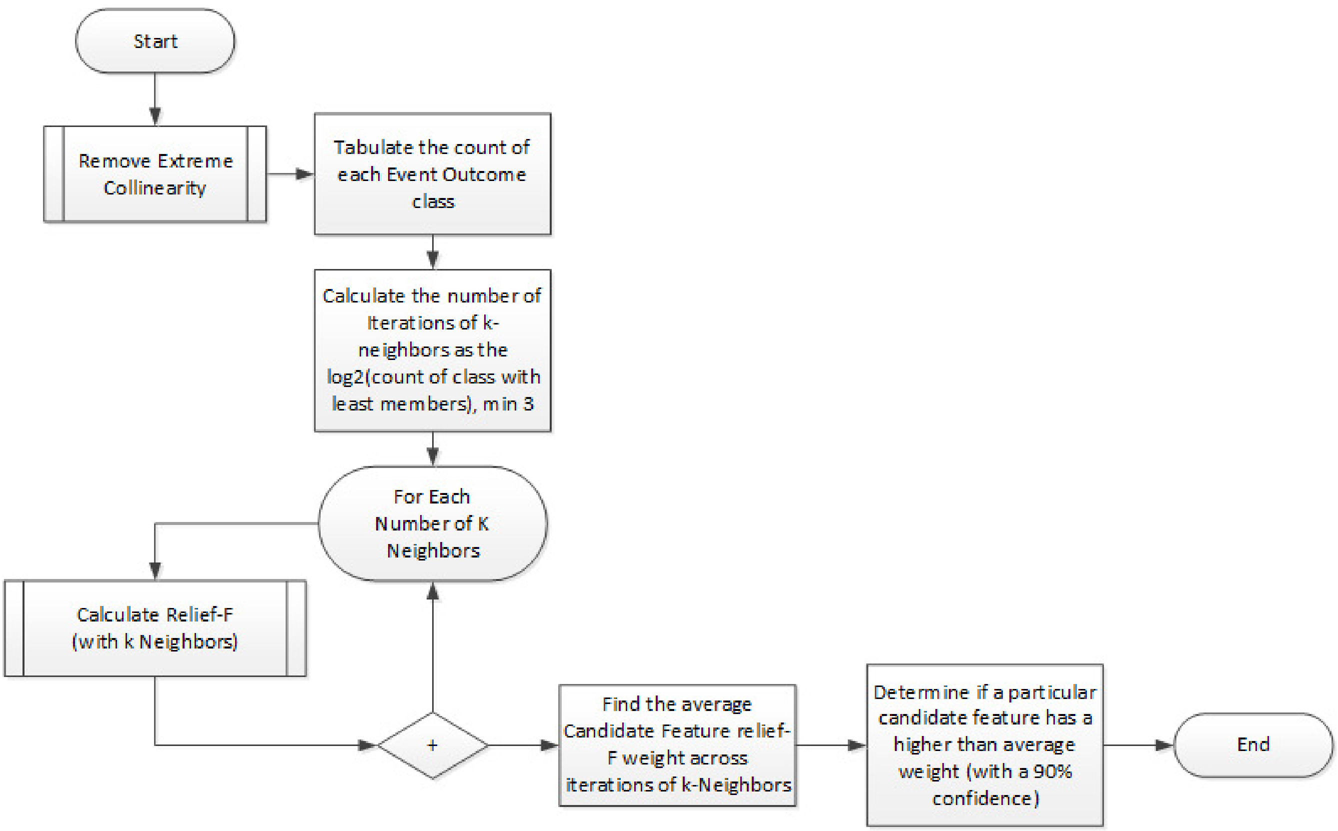


Figure 24. Multicollinearity removed, relief-f process diagram.

Approach 4: Correlation-based feature selection

Hall (1999) claimed feature selection for classification tasks in machine learning can be accomplished on the basis of correlation between features and that such a feature selection procedure can be beneficial to common machine learning algorithms. The following formula is the key component of CFS:

$$r_{zc} = \frac{k \cdot \overline{r_{zi}}}{\sqrt{k + k(k-1) \cdot \overline{r_{ii}}}}$$

where r_{zc} is the correlation between the summed components and the response variable (referred to as feature merit), k is the number of candidate features contained in the feature set, $r_{zi\text{-bar}}$ is the average of the correlations between the candidate features and the response variable, and $r_{ii\text{-bar}}$ is the average inter-correlation between the candidate features. This equation is similar to the Pearson's correlation coefficient (Anderson et al., 2005, pp. 110) with all variables standardized. In this research, Pearson's correlations were used for r_{zi} and r_{ii}, but the CFS algorithm did not necessarily require Pearson's correlation coefficient or Spearman's rank (rho; Anderson et al., 2005, pp. 850).

The central hypothesis of Hall's (1999) research, and the development of CFS, is that a good feature set contains features that are highly correlated with the class (response variable), yet uncorrelated with each other. Hall demonstrated that in most cases classification accuracy using the reduced feature set identified by CFS equalled or provided better accuracy than using the complete feature set. Hall did recognize that feature selection degraded machine learning performance in cases in which some features were eliminated because they had highly predictive power within some very small areas of the instance space (rare cases).

For this research, CFS was implemented in MathWorks® (2014) MATLAB based on the original open source code published by Hall (1999) within Weka (University of Waikato, n.d.). Weka was used to verify the MATLAB implementation was correct. A slight modification to the algorithm was made such that the CFS algorithm needed to select a minimum of two features into the features set. In rare cases when testing, the CFS algorithm yielded only one selected feature, which did not work with some of the ensemble machine learning approaches utilized in this research.

Define models [5.0]

Supervised machine learning methods utilized in research

Many supervised machine learning methods can be used for regression and classification tasks. Classification attempts to label a categorical response variable, while regression attempts to predict a continuous response variable. This research attempted to predict a discrete event outcome class label and, therefore, focused on classification models.

This research utilized three common supervised learning algorithms to classify the outcome of an event, specifically Decision Trees (DT), K-Nearest Neighbour (KNN), and Support Vector Machines (SVM). A collection of classifiers formed an ensemble. Three different ensemble types were investigated. Classification trees were considered in two different ensembles types, and one ensemble type was based on KNN. SVMs were used as a benchmark comparison to the performance of the ensemble models.

Decision trees

DTs attempt to partition the data set into smaller, more homogeneous groups. Homogeneity, in this context, means that the nodes of the split contain a larger proportion of one class in each branch (Hssina, Merbouha, Ezzikouri, & Erritali, 2014; Kuhn & Johnson, 2013, p. 370).

Tree-based models are a popular modelling method for a number of reasons. First, DTs generate a set of conditions or decision rules that are intuitive and easily understood.

Second, DTs can effectively handle multidimensional data with predictors that have different data types and distributions (i.e., skewed, continuous, categorical, etc.). Third, DTs do not require the user to specify predictors' relationship to the response variable (e.g., linear or a particular type of non-linear). Lastly, the construction of DTs does not require domain specific knowledge or parameter setting (Han et al., 2012, p. 331; Kuhn & Johnson, 2013, p. 174).

One of the primary disadvantages of trees is their instability. The performance of a single DT used for classification strongly depends on the order that features were selected. The explanation for a varying result is based on the fact that the DT creation process is known to be an unstable process. The term unstable refers to slight changes to the training data that can easily result in a different feature being chosen at a particular node in the tree, thereby resulting in significant changes in the structure of the subtree beneath that node (Witten et al., 2011, p. 352; Hastie, Tibshirani, R& Friedman, 2011, p. 312).

Decision Tree implemented parameters

Trees grown within both the bagging and boosting ensemble models are built using CART (binary splits) and employ the Gini Index as the split criteria. The advantage of the Gini split criterion over information gain split criterion is the Gini Index grows tall trees, which provides diversity within the ensemble. Diversity within the ensemble is a favourable characteristic, which effectively turns the instability of classification trees, arguably one of its greatest weaknesses as an individual classifier, into a strength. Moreover, using the Gini index can be easily extended to include costs and is computed more rapidly than information gain. For further details on the Gini index, refer to Han et al.'s work (2012, pp. 341–342).

In this research, as candidate features were derived directly from the financial security's time series pricing data, there were no missing data in the sample; therefore, the use of surrogate splits to improve the accuracy of predictions for data with missing values was not considered or required. No trees were subjected to a pruning procedure, nor were branches of leaves merged together. Candidate Features available to the DT classifier were the feature set outputted from the particular feature selection approach.

K-nearest neighbour

The KNN is a simple but effective classification technique requiring little or no prior knowledge about the distribution of the data (Imandoust & Bolandraftar, 2013). The KNN approach predicts new cases using the K-closest or the most comparable cases from the training set, similar to relief-F.

The KNN has several non-tangible advantages over other classification algorithms, including simplicity, effectiveness, and intuitiveness. It can be competitive in performance with other classification techniques used in many domains. KNN can be robust to noisy training data depending on the appropriate selection of K neighbours, and is effective if the training data are large in terms of number of training cases/tuples available for consideration (Imandoust & Bolandraftar, 2013).

The KNN has a few limitations. It can have poor run-time performance when the training set has a large number of features. Furthermore, and most relevant to this research, KNN can be very sensitive to irrelevant or redundant features because all features contribute to the similarity or distance metric between cases and thus to the classification (Imandoust & Bolandraftar, 2013).

The key elements of the KNN classifiers are the set of labelled cases, a distance or similarity metric used to compute distance between cases, and the number of nearest neighbours to consider (value of K). To classify an unlabelled new case, the distance of the new cases to the labelled training cases is computed, its KNNs identified, and the class labels of these nearest neighbours are then used to determine the class label of the new case. In the case of classification, once the nearest-neighbour list is obtained, the new case is classified based on the majority class of its nearest neighbours.

KNN implemented parameters

In this research, the KNN classifiers within the ensembles used the Manhattan (or city block) distance metric, which is equivalent to the Minkowski distance with q = 1. This metric was chosen because it is able to address both binary and multi-class classification and is arguably less sensitive to noisy data. Furthermore, the more common Euclidean may not be the best choice for high dimensional data (Aggarwal, Hinneburg, & Keim, 2001).

The number of neighbours was determined to be $Log_2(n)$, where n is the number of members in the lowest frequency class. Candidate Features available to the KNN classifier were the feature set outputted from a particular feature selection approach. All features selection approaches, with the exception of “all features”, addressed correlation among the candidate features. Furthermore, all candidate features were standardized as discussed in the Data normalization (pre-processing) section.

Support vector machine

A support vector machine (SVM) is a supervised machine-learning algorithm that performs binary classification by transforming original training data into higher dimensions to find a separating hyperplane (Han et al., 2012, p. 408). SVMs are theoretically well-founded, known to be successful in practical applications, and can be extended to multi-class classification through error correcting output code (ECOC) models.

SVM performs classification by constructing an *n*-dimensional hyperplane that optimally separates the data into the two classes, or, more specifically, finds the hyperplane maximizing the margin that separates the two classes. The distance between the derived separating hyperplane (line) within the *n*-dimensional space and the closest data points is referred to as the margin. The "best" hyperplane for an SVM is the one with the largest (maximized) margin between the two classes that has no interior data points (MathWorks®, 2014).

In some cases, it is not possible to divide the classes perfectly by a separating hyperplane. In order to handle data that are not perfectly separable, the objective function remains the same, but the decision boundary constraints need to be relaxed to allow for misclassified points. This is done by introducing slack variables into the optimization problem. This change allows some points in the training data to violate the separating line, commonly referred to as a soft-margin SVM (Fletcher, 2008).

The box constraint (C) keeps the allowable values of the Lagrange multipliers in a bounded region (MathWorks®, 2014). For small values of C, the boundary constraint would be lax, likely resulting in a wider margin. If data were not easily separable, it is likely more points would be inside of the margin, but possibly utilizing more support vectors. In contrast, if the values of C were large, the constraint would be more rigid as misclassified points are severely punished. If the data were not easily spreadable, it is likely a small margin would result with very few support vectors.

The selection of an appropriate SVM kernel function is important since the kernel function defines the feature space. Intuitively, non-linear kernels may lead to more accurate classifiers as it allows for hyperplanes that separate the classes to be curved or even more complex. However, if the number of features is large, it may not be required to map data to a higher dimensional space. That is, the non-linear mapping does not improve the performance (Hsu, Chang, & Lin, 2016). The use of a linear kernel is appropriate when the number of features is larger than the number of observations. The use of a polynomial or Gaussian kernel would likely yield a better result when the number of observations is larger than the number of features. If there are tens of thousands or more observations, speed may also be a consideration when selecting a non-linear kernel.

Multi-class classification using SVM

Binary SVM can be extended to a multi-class classification using an error correcting output coding (ECOC) model. There were two common approaches reviewed to extending binary classification SVM to multi-class classification, namely one-versus-all and all-versus-all (Han et al., 2012, p. 430). Each of these approaches is discussed in the paragraphs that follow.

In the one-versus-all approach, given m classes, m binary classifiers are trained, one for each class. Classifier *j* is trained using tuples of class j as the positive class and the remaining tuples are trained as the negative class. The resulting SVM classifier *j* learns to return a positive value for class j and a negative value for the rest. To classify an unknown tuple, each of the trained classifiers has an equally weighted vote as an ensemble (see Table 9).

Table 9

One-versus-All Classifier Matrix

Class	SVM1	SVM2	SVM3	SVM4
Class 1: Excellent	+1	-1	-1	-1
Class 2: Favourable	-1	+1	-1	-1
Class 3: Unfavourable	-1	-1	+1	-1
Class 4: Terrible	-1	-1	-1	+1

Note. SVM = Support Vector Machine.

All-versus-all is an approach in which classifiers are trained in pairs of classes. Given m classes, m(m-1)/2 binary classifiers are trained using only tuples of the two classes. For each binary learner, one class is positive, another is negative, and the rest are ignored. This design exhausts all combinations of class pair assignments. To classify an unknown tuple, each classifier votes. In the example in Table 10, SVM1 trains on observations having Class 1 and Class 2, treating Class 1 as positive (+1) and Class 2 as negative (-1), all other tuples are excluded (0).

Table 10

All-versus-All Classifier Matrix

Class	SVM1	SVM2	SVM3	SVM4	SVM5	SVM6
Class 1: Excellent	+1	+1	+1	0	0	0
Class 2: Favourable	-1	0	0	+1	+1	0
Class 3: Unfavourable	0	-1	0	-1	0	+1
Class 4: Terrible	0	0	-1	0	-1	-1

Note. SVM = Support Vector Machine.

SVM implemented parameters

Given the number of candidate features generated in this research, a linear kernel function was selected. Furthermore, the researcher selected a linear SVM as it was considered less prone to overfitting than non-linear. Most of feature selection approaches within this research yielded more candidate features than training tuples, further justifying the linear separation. Given this selection, it is possible that feature selection approaches that resulted in fewer features than training tuples would not perform as well as those feature selection approaches that yielded a feature set with a count of features greater than the number of training tuples.

The box constraint was set to C = 1 to allow for a relaxed boundary constraint. Given the nature of the research, some event outcomes were driven by external factors not reflected by the technical indicators; therefore, the event outcomes were not linearly separable in all cases and some level of flexibility was required to find a feasible solution.

To extend the binary SVM model to a multi-class problem, an all-versus-all approach for ECOC model was selected. All-versus-all tends to be superior to one-versus-all (Han et al., 2012, p. 431).

Classification ensemble models

An ensemble is a collection of classification models, or classifiers, that combine the decisions of the individual classifiers into a single prediction. The simplest technique to amalgamate the individual predictions of each ensemble member in the case of classification is to take an equal or weighted vote (Witten et al., 2011, p. 352).

The use of ensemble techniques has been proven, both theoretically and empirically, to outperform a single prediction model approach (Aly & Atiya, 2006; Han et al., 2012, p. 377). Ensembles yield better results when there is significant diversity among the models (Han et al., 2012, p. 378). A good ensemble is one in which the individual members of the ensemble are both accurate, but make errors in different areas of the input space (Opitz & Maclin, 1999). Ensemble learning in general can improve predictive performance by increasing the diversity of the classifiers contained in the ensemble and taking advantage of instability. Neural nets, classification trees, and regression trees are generally considered unstable procedures (Breiman, 1996; Opitz & Maclin, 1999). Bagging (Breiman, 1996), boosting (Freund & Schapire, 1996) and random forest (Breiman, 2001) are examples of popular ensemble methods used in this research.

Bagging

Bagging is an ensemble method for generating multiple different classification trees into a collection to create an aggregated prediction to counterbalance the instability of classification trees (Witten et al., 2011, p. 354). A bagged ensemble often has significantly greater accuracy than a single classifier, or at the very least, it will not be considerably worse and is more robust to the effects of noisy data and overfitting. The increased accuracy occurs as the ensemble reduces the variance of any one individual classifier (Han et al., 2012, pp. 379–380; Witten et al., 2011, p. 354).

The term bagging stands for bootstrap aggregation (Breiman, 1996). Instead of obtaining independent data sets, bagging resamples the training data multiple times. Each classifier's training set is generated by randomly selecting, with replacement, *n* tuples (events). Many of the tuples may be repeated in the resulting training sets while others may be left out. As a result, each individual classifier in the ensemble are generated with a different random sample of tuples from the full training set (Opitz & Maclin, 1999; Witten et al., 2011, p. 354). An algorithm for bagging can be found in Han et al.'s (2012, p. 380) work.

Random forest

A random forest is a specific type of bagging ensemble. A random forest is an ensemble of classification trees generated from bootstrap sampling the training data (bagging), but with the addition of every tree in the ensemble also being grown by randomly selecting features for decision splits in the tree creating process (Breiman, 2001). Similar to bagging, features are made by aggregating (using majority vote for classification) the predictions of each ensemble member (MathWorks®, 2014).

Random forests change how the classification trees are constructed. In standard trees, each node is split using the best split among all features. In a random forest, each node is split using the best feature among a subset of features randomly chosen at that node (Liaw & Wiener, 2002). The accuracy of a random forest depends on the strength of the individual tree classifiers and sufficient independence or dissimilarity between the trees (Breiman, 2001; Ho, 1998). Random forests are comparable in accuracy to adaptive boosting or AdaBoost (discussed in "Boosting" section later in this chapter), yet are more robust to errors and outliers. Random forests can create an effective predictor with minimal risk of overfitting as a result of randomness and the law of large numbers (Breiman, 2001). The generalization error for a forest converges as long as the number of trees in the forest is sufficiently large (Han et al., 2012, p. 383). In addition to being comparable in accuracy to AdaBoost and a having a greater resilience to outliers and noise, random forests are also generally faster than bagging or boosting as they can be easily parallelized (Breiman, 2001). A more technical overview of the random forest algorithm can be found in Hastie et al.'s (2011, p. 588) work.

Number of variables to select

An important parameter is the number of features selected at random for every decision split within random forest ensembles. It is ideal to maintain the strength of individual classifiers without increasing their correlation. Usually, selecting a small number of features for consideration at a decision node gives near optimum results as selecting larger numbers of features begins to increase the correlation among the ensemble members (Breiman, 2001). Han et al. (2012, p. 383) recommend selecting $\log_2 N + 1$ features. If the input space has a very large number of variables but expects only very few to be important, using larger number of candidate features for the split at a node may give better performance (Liaw & Wiener, 2002).

Class imbalance and random forest

Similar to most classifiers, the performance of random forests can suffer from a highly imbalanced training data set. Random forests are constructed to minimize the overall error rate; therefore, a random forest will tend to focus more on the prediction accuracy of the majority class, which often results in poor accuracy for the minority class (Chen, Liaw, & Breiman, 2004). Chen et al. (2004) proposed two approaches to deal with the problem of imbalance data, both based on the random forest algorithm (Breiman, 2001). One approach incorporates class weights into the random forest classifier, making it cost sensitive and penalizing to misclassifying the minority class. The other approach combines sampling techniques and the ensemble idea by down-sampling the majority class and growing each tree on a more balanced data set. A majority vote is taken as usual for prediction.

Random forest ensemble implemented parameters

Although $\log_2 N + 1$ is a common parameter setting in the literature, Liaw and Wiener (2002) provided a convincing argument that high-dimensional input spaces with only a few features expected to be important should include a larger number of candidate features at each decision point. Therefore, the number of predictors selected at random for each split in this research was set to the square root of N, where N was the number of candidate features contained in the feature set, rounded up to the nearest whole number.

To address class imbalance, this research uniformly applied a cost matrix to each ensemble method considered, which is discussed in the "Data sampling and cost matrix" section later in this chapter. If the misclassification cost is highly imbalanced, then, for in-bag samples associated with random forests, MATLAB oversamples unique observations from the class that has a large penalty (MathWorks®, 2014).

Boosting

Ideally, each classifier within the ensemble should complement one another, with each classifier being a specialist in a part of the problem domain in which the other learners do not perform very well (Witten et al., 2011, pp. 357–358). Boosting, generally considered to have been pioneered by Freund and Schapire (1996), is a sequential algorithm in which each classifier in the ensemble is constructed with consideration of the performance of previously generated classifiers with the goal of combining classifiers that better complement one another (Martinez-Munoz & Suarez, 2007). Similar to bagging, boosting uses voting to combine the output of the individual classifiers.

In boosting, tuples that are incorrectly predicted by classifiers previously added to the ensemble are sampled more frequently to train the next classifiers. Consequently, tuples that were predicted correctly by existing classifiers in the ensemble are selected less frequently. The objective of boosting is to produce new classifiers that are better able to predict tuples for which the current ensemble members have a poor performance (Opitz & Maclin, 1999). Opitz and Maclin (1999) argued some of the increases in performance produced by boosting are dependent on the particular characteristics of the data set rather than on the component classifiers. As the noise level grows, the efficiency of the bagging ensembles generally increases, while some boosting ensembles experience marginally smaller performance gains or potentially decrease in performance (Opitz & Maclin, 1999). In contrast to boosting, the resampling of the training set in bagging is not dependent on the performance of the earlier classifiers. Using a bagging approach, individual models are built separately, but in boosting each new classifier is influenced by the performance of the classifiers previously built; as such, boosting is sequential and more difficult to parallelize.

Additive models, such as boosting, generally focus on the misclassified tuples, which introduces the risk of overfitting (Freund & Schapire, 1996; Han et al., 2012; Opitz & Maclin, 1999). The bagging method is less susceptible to model overfitting, as it treats each tuple with equal importance. One approach to decrease the risk of overfitting the model is to use shallow trees, as opposed to bagging in which generally deeper trees are desirable (MathWorks®, 2014).

AdaBoost

The most common boosting algorithm is AdaBoost (or adaptive boost) proposed by Schapire and Freund (as cited in Seiffert, Khoshgoftaar, Van Hulse, & Napolitano, 2008). The AdaBoost algorithm became popular with a strong theoretical foundation and a reputation for accurate predictions in a variety of applications (Wu et al., 2008). In 2003, Schapire and Freund (as cited in Wu et al., 2008) won the Gödel Prize for their AdaBoost paper.

Initially, AdaBoost assigns each training tuple an equal weight. In iteration *i*, the tuples from the full training data set *D* are sampled to form a training subset of d_i. Sampling with replacement is used, but each tuple's chance of being selected as a sample is based on its weight. Based on the error of the classifier, the weights of the tuples are updated to allow a subsequent classifier, M_{i+1}, to more actively select the training tuples that were misclassified by M_i. If a tuple was incorrectly classified, its weight is increased. If a tuple was correctly classified, its weight is decreased. A tuple's weight (w_j) reflects how difficult it is to classify—higher weight indicates the tuple is more often misclassified. To compute the error rate of model M_i, sum the weights of each of the tuples in D_i that M_i misclassified. That is,

$$error(M_i) = \sum_{j=1}^{d} w_j \times err(X_j)$$

where $err(X_j)$ is the misclassification error of tuple X_j. If the tuple was misclassified, then $err(X_j)$ is 1; otherwise, it is 0. The lower a classifier's error rate, the more accurate it is, and, therefore, the higher its weight for voting should be. The weight of classifier M_i 's vote is

$$W_i = \log \frac{1 - error(M_i)}{error(M_i)}$$

The final boosted classifier, M*, combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy (Han et al., 2012, p. 380). A detailed technical algorithm is found in the works of Han et al. (2012, p. 382), Wu et al. (2008), as well as Schapire and Freund (as cited in Seiffert et al., 2008).

AdaBoostM2, which is an extension of AdaBoostM1, can be used for multiple classes. Instead of weighted classification error, AdaBoostM2 uses weighted pseudo-loss for *d* observations and *C* classes. Pseudo-loss can be used as a measure of the classification accuracy from any learner in an ensemble. Pseudo-loss typically shows the same behaviour as a weighted classification error for AdaBoostM1 (MathWorks®, 2014).

Random under sampling boost

Random under sampling boost (RUSBoost) is a boosting algorithm designed to handle class imbalance in data with discrete class labels. It uses a combination of random under sampling and the standard boosting procedure AdaBoost to better model the minority class by removing majority class samples (Seiffert et al., 2008). The algorithm takes the class with the fewest members in the training data as the basic unit for sampling. For each classifier in the ensemble, RUSBoost takes a subset of the data from each of the majority classes. The boosting procedure follows the procedure in AdaBoostM2 for reweighting and constructing the ensemble (MathWorks®, 2014).

The main drawback of random under sampling is the loss of information. This disadvantage is in theory overcome with boosting. While certain information may be missing during the construction of one of the ensemble members, the tuple is likely to be included in construction of another within the boosting ensemble. Oversampling does not result in the loss of information, but can lead to overfitting (Seiffert et al., 2008).

Boosting ensemble implemented parameters

In this research, all trees contained in an ensemble using the boosting method were built using CART and the Gini Index as the split criteria. In contrast to a bagging ensemble, in which tall trees consisting of multiple splits are used to introduce diversity, trees in the boost ensemble were designed to consist of only a single split. A tree with only a single split is referred to as a Decision stump. Decision stumps are known to work well in boosting ensembles as it supports further diversification among the ensemble methods and decreases the risk of overfitting. A decision stump is one root node connected to two terminal leaf nodes.

The boosting ensemble is implemented using RUSBoost algorithm as described. Similar to bagging, all training data were resampled with replacement to train the learners in the ensemble; although, different from bagging, the sequential nature of the boosting algorithm considered all training tuples available for selection by every classifier based on the tuple weight.

Subspace

The KNN algorithm is a stable learner in comparison to DTs and other supervised learning algorithms such as neural nets (Breiman, 1996). The use of ensembles to train a collection of KNN classifiers can improve the performance as long as diversity can be introduced into the ensemble (Brofos, 2014).

To introduce diversity into KNN models, features are randomly selected for each KNN classifier in the ensemble. KNN classifier's prediction depends on the distances between tuples, which in turn depends on the selected features. As a result, KNN classifiers can become unstable learners by randomly selecting subsets of the feature set (Witten et al., 2011, p. 357). This approach is known as the random subspace method for constructing an ensemble of multiple classifiers. Random subspace feature selection ensembles can improve the accuracy of KNN classifiers (MathWorks®, 2014).

KNN ensemble implemented parameters

The number of predictors to sample for each random subspace learner was set equal to Log_2N, where N is the number of candidate features contained in the feature set, rounded up to the nearest whole number. The number of K neighbours was set equal to Log_2R, where R is the number of observations of the rarest class. All tuples were considered when developing each ensemble member.

General ensemble parameters and considerations

There are several parameters and general considerations that were common among all ensemble methods in this research. The common parameters included the number of classifiers/ensemble members, the management of imbalanced data (rare event outcomes), and the homogeneity in terms of the type of learner contained.

Number of classifiers/members contained in an ensemble

Opitz and Maclin (1999) compared DT and NN ensembles to investigate the performance improvements relative to the number of members contained in the ensemble. The largest marginal reduction in error occurs within the first 10 to 15 members of the ensemble and relatively large gains observed for the first 25 members with boosting and decision trees (Opitz & Maclin, 1999). The number of members necessary for good performance grows with the number of features (Liaw & Wiener, 2002).

An ensemble can require from a few dozen to a few thousand weak learners depending on a variety of factors including the number of features, the configuration of the classifiers used in the ensemble, the selected ensemble method, and the number of tuples contained in the data set. As it is difficult to individually evaluate the number of appropriate members for each individual ensemble model considered across the wide variety of profiles, the number of weak classifiers (members) contained in each ensemble was set at a constant 300. This value was selected in an attempt to find a balance between computation time and marginal information gain.

Combining different types of classifiers into an ensemble

Ensembles are generally homogenous, meaning all members are the same type of classifier (e.g., contains all trees). Tsai et al. (2011) investigated the performance of ensembles to analyze the quarterly rate of return for stocks in the electronic industry, specifically using homogeneous classifier ensembles (e.g., an ensemble of NNs) in comparison to heterogeneous classifier ensembles (e.g., an ensemble of NNs, DTs, and logistic regression). The best model presented was based on a homogeneous ensemble that used a majority vote. Within this research, each ensemble model was a collection of homogenous classifiers.

Class imbalance

When modelling discrete classes, the relative frequencies of each class can have an impact on the performance of the model. Class imbalance occurs when one or more classes are observed far less frequently in relative comparison to the other classes (Kuhn & Johnson, 2013, p. 419).

Traditional classification algorithms aim to minimize the number of errors made during classification, resulting in the assumption that the costs of a false positive or a false negative error are equal among all classes (Han et al., 2012, p. 384). In most real-world applications, it is the rare (underrepresented or minority) classes that generally carry the highest cost of misclassification (Seiffert et al., 2008). Two common techniques for addressing class imbalance are data sampling and cost-sensitive learning, each of which is discussed in the paragraphs that follow.

Data sampling and cost matrix

Data sampling techniques balance the class distribution of the training data by either adding examples to the minority class (oversampling) or removing examples from the majority class (undersampling). Oversampling works by resampling such that the rare classes contain more samples in the training set. Undersampling works by decreasing the number of common samples until there is a more equal number of tuples from each class (Han et al., 2012, p. 384).

Synthetic minority oversampling technique (SMOTE) is a data sampling procedure that uses both oversampling and undersampling. To oversample the minority class, SMOTE synthesizes new events by randomly selecting a data point from the minority class and then selecting the KNN from this point (Chawla, Bowyer, Hall, & Kegelmeyer, 2002; Kuhn & Johnson, 2013; Weiss, 2004). Chawla et al. (2002) used five neighbours in their analyses, but different values can be used depending on the data. The new synthetic tuple is a combination of the randomly selected KNN tuples. This approach effectively forces the decision region of the minority class to become more general (Chawla et al., 2002).

The second of the two common techniques for addressing class imbalance is cost-sensitive learning. Many of the predictive models for classification have the ability to use weighting in which each individual data point can be given more emphasis in the model training phase (Kuhn & Johnson, 2013, p. 426). Some models use prior probabilities from the training data to determine cost-weights unless specified manually. In general, manipulating the cost matrix is equivalent to manipulating the prior probabilities. If you have three or more classes, it is possible to convert input costs into adjusted prior probabilities (MathWorks®, 2014)

Implemented parameters

Classification techniques usually do not have an ordinal component. For example, if classifying different types of fish, a salmon is not better than a bass—they are just different fish. Within this research, it seemed prudent to account for the fact that classifying an event outcome as favourable, when the actual outcome is excellent, is different than classifying an event outcome as excellent, when the actual outcome is terrible. Furthermore, this research had the challenge that each profile created had a different set of events that were unique to the combination of the financial security and trading system represented. To address this characteristic of the data a dynamic cost matrix approach was created.

This research implemented a cost-sensitive learning approach to address the imbalance of event outcomes classes (e.g., extremely profitable events are rare). It is common to introduce a cost matrix in which misclassification of one class is more important than others. The MATLAB (MathWorks®, 2014) application developed in support of this research used the cost matrix to adjust the prior class probabilities, and then used the adjusted prior probabilities for training, and reset the cost matrix to its default. The cost matrix is square. Cost(i,j) is the cost of classifying an event into class j, if its true class is *i*. That is, the rows correspond to the true class and the columns correspond to the predicted class.

To specify a cost matrix for each profile, the Euclidean distance was used to represent ordinal aspects of the different classes. For example, the distance between the bin centre of excellent and terrible will be greater than the distance between the event outcomes excellent and favourable. In

Table 11, the example bin labels and means are shown on the left. On the right is the corresponding example cost matrix. The cost of predicting an event as excellent, when the actual event outcome label is terrible, is calculated as Sqrt[(-0.04 – 0.06)^2] = 0.10. As a result, the application oversamples classes with larger misclassification costs and under samples classes with smaller misclassification costs.

Table 11

Example Cost Matrix

		Cost Matrix		
Event Outcomes	Terrible	Unfavourable	Favourable	Excellent
Terrible Bin Mean: -4.0%	-	0.03	0.05	0.10
Unfavourable Bin Mean: -1.0%	0.03	-	0.02	0.07
Favourable Bin Mean: 1.0%	0.05	0.02	-	0.05
Excellent Bin Mean 6.0%	0.10	0.07	0.05	-

Summary of ensemble properties and implementation

All ensembles in this research were created using the common parameters summarized in Table 12.

Table 12

Summary of Ensemble Properties and Implementation

Model Parameter	Description	Input Variable	Source in Application
X	Each row of X corresponds to an event, and each column corresponds to a candidate feature value.	Feature Set	The output of the feature selection process.
Y	Y is a column vector containing the categorical label of the event outcome with a length equal to the number of events in X.	Response Variable	The output is based on the selected response mode (multi-classification or binary) and the response type (binning or clustering).
N Members	The number of ensemble learners/members contained in the ensemble.	300	Constant. All ensembles are constructed with 300 classifiers.
Ensemble Method	The method used to create the ensemble model.	Bagging (RF), subspace, or boosting (RUS)	Function of research design. See Table 13.
Classifier Type	The type of learner	Tree or KNN	Function of research design. See Table 13.
Cost Matrix	Misclassification cost for class imbalance	A matrix (*i,j*) that defines the cost of classifying a point into class *j* if its true class is *i*.	A cost matrix based on distance to class outcome centre.
Prior	Empirical distribution, reflects historic event outcome frequency.	The class prior probabilities are based on the event outcome relative frequencies in Y.	Based on the distribution of event outcome classification labels of each profile.

Note. KNN = K-Nearest Neighbour; RF = Random Forest; RUS = Random Under Sampling.

Summary of ensemble methods and classifier types

Table 12 provides a summary of the ensemble methods and classifier types used in this research. The table shows the treatment of events and candidate features contained in the feature set resulting from a specific feature selection approach.

Table 13

Summary of Ensemble Methods and Classifier Types

Ensemble Type	Classifier/ Learner	Treatment of Events (Rows)	Treatment of Features (Columns)
Bagging: Random Forest	Tree	Randomly selected for each member	Randomly selected for each split
Boosting: Random Under Sampling	Tree	Randomly selected; events are weighted based on difficultly to classify. Therefore, events that are harder to classify are sampled more frequently	All candidate features considered at each split
Subspace	KNN	All events considered for each member	Randomly selected for each member

Note. KNN = K-Nearest Neighbour.

Generate models [5.0.1]

Based on the above defined ensemble models and four feature selection approaches there were a total of 16 ensemble models generated for each profile. An ensemble model is created for each feature selection approach. The feature set contains a collection of candidate features. The response variable (Y) is then assigned a categorical label associated with the response variable mode. *Figure 25* provides a visual depiction of the process diagram for model generation.

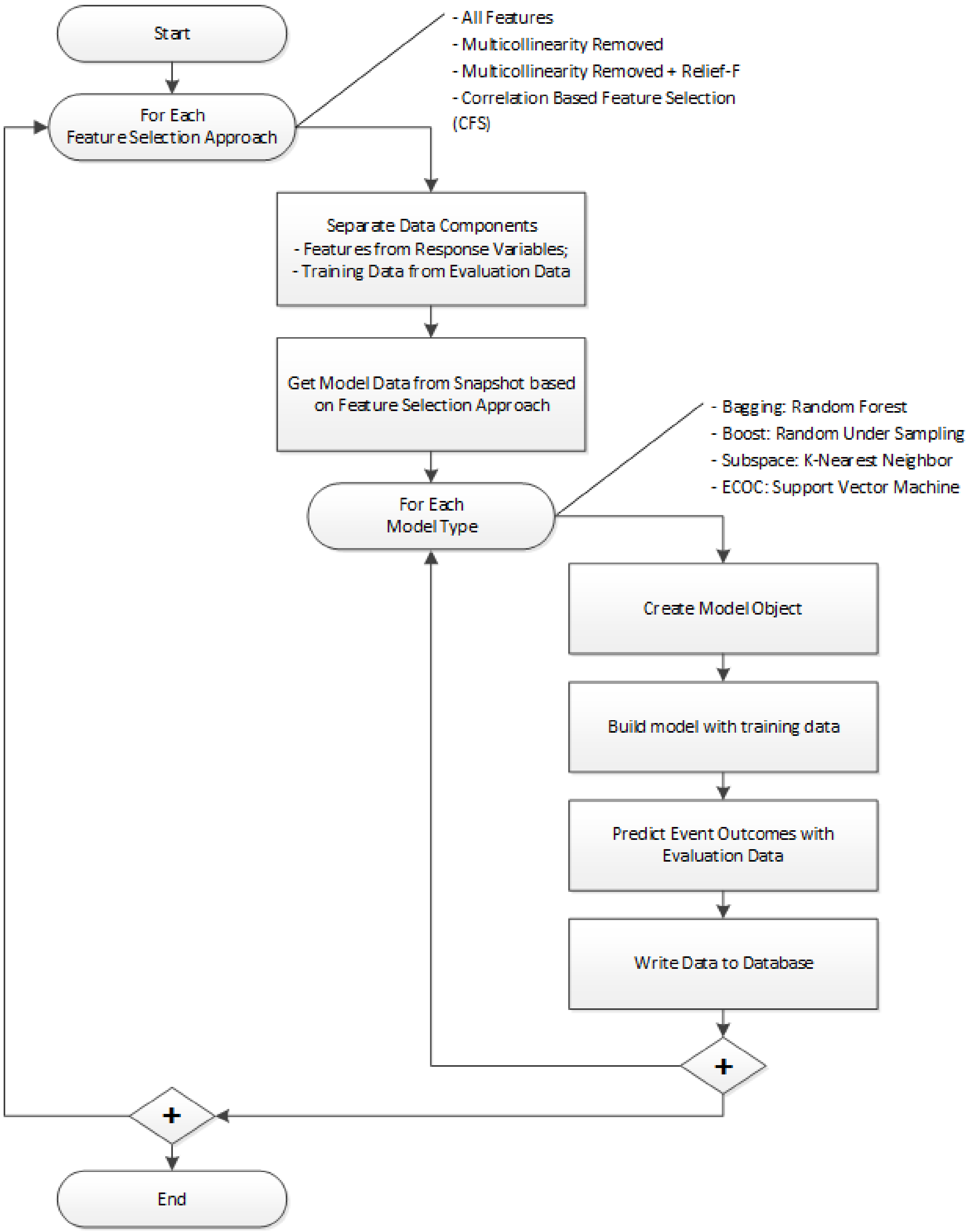


Figure 25. Generate model process diagram.

Note. CFS = Correlation-Based Feature Selection; ECOC = Error Correcting Output Code; KNN = K-Nearest Neighbour.

Evaluating performance [6.0]

Response mode scenarios

This research inherently assumes that both the training and the evaluation data contained a representative sample of future events. This research used an exploratory data set consisting of 30 financial securities to investigate the high-level impacts of clustering versus binning events as well as exploring the performance of multi-class versus binary classification. Based on the preliminary results of that exploratory data set (presented in Chapter 5) this research examined a second, larger, independent set consisting of 90 financial securities for the response mode 'multi-class clustering' only. The objective of a second, larger data set was to expand both the number of profiles and events, thereby increasing the sample sizes.

Partitioning data for model evaluation

It is generally considered bad practice to evaluate the performance of ensembles classification models based on training data, as doing so tends to produce overly optimistic estimates of their classification abilities (Brofos, 2013; MathWorks®, 2014). One approach to partitioning data into training and evaluation sets is the holdout method, which sets aside a certain portion of the data for training and uses the remaining data for testing. It is common to hold out approximately one-third of the data for testing and use the remaining two-thirds for training (Witten et al., 2011, p. 149; see *Figure 26*). The data contained in the snapshot is partitioned into model training data and model evaluation data, with 30% of the events for a given profile remaining unseen by the training set and then used for evaluation.

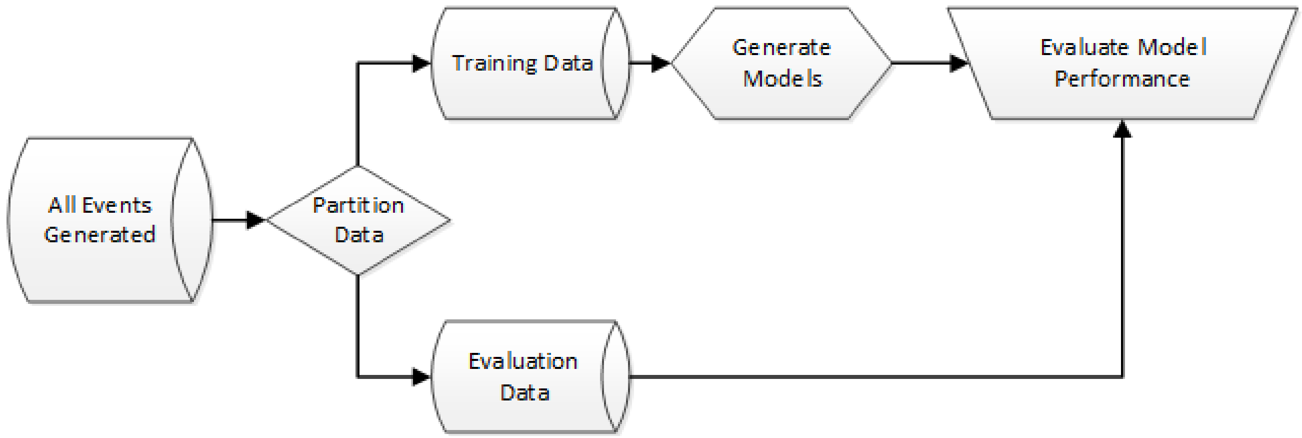


Figure 26. High-level analysis process diagram.

A walk-forward approach was selected to reflect practical application. In this research, the events in the profile were divided into the training set (70%) and evaluation set (30%). The research used data from the years 2002 to 2016 inclusive (20 years of data). The training period was approximately 2002 to 2012, and the evaluation period was approximately 2012 to 2016, depending on the number of events generated in a particular profile[3].

An alternative method considered was to randomly sample using *k-fold cross-validation*. In k-fold cross-validation, the initial data are randomly partitioned into k mutually exclusive subsets or folds, D_1, D_2, …, D_k, with each fold being approximately equal in size. Training and testing is performed k times. In iteration *i*, partition D_i is reserved as the test set, and the remaining partitions are collectively used to train the model. For example, a model is fit using all of the samples except the first subset (called the first fold). The held-out samples are predicted by this model and used to estimate performance measures. The first subset is returned to the training set and the procedure

[3] Actual division is based on proportion of events in the profile, not a date range

repeats with the second subset held out, and so on. Each sample is used the same number of times for training and once for testing (Han et al., 2012, p. 370; Kuhn & Johnson, 2013, pp. 69–70). This method was not used in this research, as the walk-forward method was considered to better reflect real-life application.

Confusion matrix

The confusion matrix is commonly used for analyzing the performance of classification models. True positives (TPs) and true negatives (TNs) show when the classification model is correctly identifying cases of different classes, while false positives (FPs) and false negatives (FNs) show when the classification model is incorrectly identifying cases of different classes (Han et al., 2012, p. 365). *Figure 27* provides an example of a binary confusion matrix.

		Actual	
		Yes	*No*
Predicted	*Yes*	**True Positive (TP)**	False Negative (FN)
	No	False Positive (FP)	**True Negative (TN)**

Figure 27. Binary confusion matrix.

In multi-class prediction, the results are displayed as a two-dimensional confusion matrix with a row and column for each class. Good results correspond to large numbers down the main diagonal and small (ideally zero) off-diagonal elements (Witten et al., 2011, p. 164). For example, *Figure 28* presents the confusion matrix for the example class *undesirable*. The FP cases indicate where undesirable was predicted but did not occur, and the FN cases indicate where undesirable was not predicted but actually occurred.

		Actual					
		Excellent	*Desirable*	*Favourable*	*Unfavourable*	***Undesirable***	*Terrible*
Predicted	*Excellent*	TN	TN	TN	TN	**FN**	TN
	Desirable	TN	TN	TN	TN	**FN**	TN
	Favourable	TN	TN	TN	TN	**FN**	TN
	Unfavourable	TN	TN	TN	TN	**FN**	TN
	Undesirable	**FP**	**FP**	**FP**	**FP**	**TP**	**FP**
	Terrible	TN	TN	TN	TN	**FN**	TN

Figure 28. Multi-class confusion matrix for undesirable.
Note. FN = False negative; FP = False Positive; TN = True Negative; TP = True Positive.

Performance metrics

Several measures of classification model performance can be derived from the confusion matrix. The accuracy of a classifier is the percentage of cases that are correctly classified (Han et al., 2012, p. 366). Accuracy works best if FPs and FNs have similar costs. The formula for accuracy is as followed:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

The precision and recall measures are also widely used in classification. Precision is a measure of exactness and recall is a measure of completeness (Han et al., 2012, p. 368). These measures can be computed as followed:

$$True\ Positive\ Rate\ (Recall) = \frac{TP}{TP + FN}$$

$$Positive\ Predictive\ Value\ (Precision) = \frac{TP}{TP + FP}$$

Recall is the number of correctly classified positive examples divided by the number of actual positive examples in the data. Precision is the number of correctly classified positive examples divided by the number of examples predicted by the classification model as positive (Sokolova & Lapalme, 2009). Note that precision and recall do not depend on TN, but only on the correct labelling of positive examples (TP) and the incorrect labelling of examples (FP and FN).

Precision and recall can be combined into a single measure known as the F_{beta} measure or F-score

$$F_\beta = \frac{(1+\beta^2) \times \; Precision \; \times \; Recall}{\beta^2 \times \; Precision \; + \; Recall}$$

When β is set equal to 1, the F measure is the harmonic mean and gives equal weight to precision and recall. When β is set not equal to 1, F_{beta} measure becomes a weighted measure of precision and recall assigning more weight to recall or precision. Commonly used F_{beta} measures are F_2, which weights recall twice as much as precision and $F_{0.5}$, which weights precision twice as much as recall (Han et al., 2012, p. 369).

Class imbalance

The accuracy metric works best when the data classes are fairly evenly distributed. The issue with the accuracy metric is that the performance of rare classes is less represented than more common classes (Chawla et al., 2002; Weiss, 2004; Witten et al., 2011). The accuracy metric is not an appropriate measure for data sets with class imbalance due to the fact that a classification model that constantly predicts the majority classes and incorrectly predicts the rare classes of interest while still achieving a high accuracy. As the event outcome class distribution is skewed, an ensemble classifier can achieve a low misclassification rate by more frequently classifying the majority class. Other measures, such as recall, precision, and/or F-score are better suited to evaluate domains with class imbalance or when the main classes of interest are rare (Han et al., 2012, p. 370).

Multiple classes

There are two possible approaches to calculating performance metrics in multi-class classification, namely micro-averaging and macro-averaging. Micro-averaging is calculated by summing the counts to obtain the cumulative values of TP, TN, FP, and FN, and then calculating performance measures using these cumulative values. The overall accuracy of the classification model is an example of micro-averaging.

In data sets with class imbalance, it can be desirable to have a classifier that provides high prediction accuracy over the minority class, while maintaining a reasonable accuracy for the majority class (Chen et al., 2004). Macro-averaging is calculated by determining the performance measure for each class and then summing the performance measure values and dividing by the number of classes. Macro-averaging treats all classes equally while micro-averaging by nature favours larger classes.

Macro-averaging is an approach that allows all classes to be considered equally. The following is the macro equation for F_{beta} where C is the number of classes.

$$Macro\ F_{\beta} = \frac{1}{C}\sum_{i=1}^{C} F_{\beta_i}$$

Other macro-average classification metrics are calculated in a similar fashion by individually calculating the classification performance measure for each class, then taking the average of the metrics among the classes.

This research used the macro-harmonic mean (F_1) to evaluate if there was a difference among the classification models within the ANOVA. The harmonic mean is metric that is a better indicator for uneven class distribution than accuracy.

Accounting for random chance

The performance of the ensembles was measured using the macro-averaged approach to account for class imbalance and then used to determine if there is a difference in classification performance. The kappa statistic and Matthews correlation coefficient were used to evaluate the performance of the ensembles relative to random chance.

Kappa statistic

The kappa statistic considers the accuracy that would be achieved simply by chance. The kappa statistic is used to measure the agreement between predicted and observed cases within a data set, while correcting for an agreement that occurs by chance (Witten et al., 2011, p. 166). The kappa statistic is as follows:

$$Kappa = \frac{O - E}{1 - E}$$

where O is the observed accuracy, and E is the expected accuracy based on the marginal totals of the confusion matrix. The kappa statistic can take on values between −1 and 1; a value of 0 means there is no agreement between the observed (actual) and predicted classes, while a value of 1 indicates perfect concordance of the model prediction and the observed/actual classes. A value less than zero suggests that the classifier performs worse than random chance. Depending on the context, kappa values within 0.30 to 0.50 indicate practical agreement (Kuhn & Johnson, 2013, p. 255).

In the case of this research, any kappa value greater than zero indicated the classification model successfully extracted additional information from a financial time series pricing data and provided value beyond random chance. To support the claim the classification model performed better than random chance the model needed to have a kappa value statistically greater than zero.

Mathews correlation coefficient

The Mathews correlation coefficient (MCC) provides correlation between the observed and predicted classifications and returns a value between −1 and +1. A coefficient of +1 represents a perfect prediction, 0 indicates the predicted classifications are no better than random prediction, and −1 indicates total disagreement between prediction and actual.

$$MCC \frac{Cov(X,Y)}{\sqrt{Cov(X,X) * Cov(Y,Y)}}$$

In the above equation, Cov represents the covariance between the predicted labels and the actual labels. The MCC has been generalized to the multi-class case. This generalization is called the R_K statistic (for K different classes) and is defined in terms of a K × K confusion matrix represent by C (Jurman, Riccadonna, & Furlanello, 2012).

$$MCC \frac{\sum_{k,l,m=1}^{K} C_{kk}C_{ml} - C_{lk}C_{km}}{\sqrt{\sum_{k=1}^{K}\left[\left(\sum_{l=1}^{K} C_{lk}\right)\left(\sum_{f,g=1 f\neq k}^{K} C_{gf}\right)\right]}\sqrt{\sum_{k=1}^{K}\left[\left(\sum_{l=1}^{K} C_{kl}\right)\left(\sum_{f,g=1 f\neq k}^{K} C_{fg}\right)\right]}}$$

Similar to the standard interpretation of correlation, a positive MCC value suggests the predicted classification label aligns with the actual class labels. The degree of comparative competitive advantage is reflected by the strength of the correlation.

This research used the kappa statistic to assess if a classification model performed better than random chance. The MCC was used as a confirming measure. Logically, a classifier that performed better than random chance would also have a relatively greater correlation between the predicted labels and the actual labels in comparison to a classification model that did not perform better than random chance.

CHAPTER V: RESULTS

The cluster multi-class response mode was selected from the four response mode options considered based on the performance measures of the exploratory data sets. The evaluation data of the verification set was used to investigate the following summarized research goals, which are discussed in the subsections that follow:

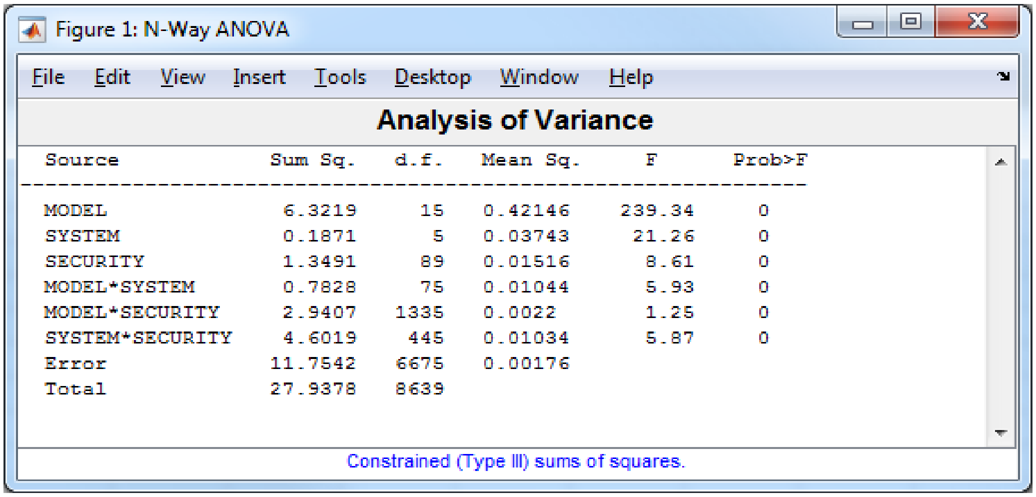
- Determine if there is a difference in the performance among the selected ensemble classification models in terms of identifying the outcome of an event;
- Determine if it is possible to develop an ensemble classification model that is able to predict the outcome of an event using a diverse set of technical market indicators with a performance level that can be considered superior to random chance; and
- Determine which classification model is comparatively superior among the models considered.

Differences in model performance

An ANOVA was performed on the evaluation data of the verification set using the macro F-Score to evaluate if a difference exists among the performance of the classification models considered. Although the F-score is difficult to interpret, aside from the harmonic mean of both precision and recall, it provides a balanced metric that can be helpful for concluding if there is a difference in the performance characteristics of classification models considered.

Analysis of Variance

Figure 29 depicts a standard ANOVA table showing the significance of each factor and the significance of the interaction among the factors in relation to model performance as measured by the F-Score. The ANOVA shows each of the three factors (security, trading system, and model) had an influence on the average F-Score.



Source	Sum Sq.	d.f.	Mean Sq.	F	Prob>F
MODEL	6.3219	15	0.42146	239.34	0
SYSTEM	0.1871	5	0.03743	21.26	0
SECURITY	1.3491	89	0.01516	8.61	0
MODEL*SYSTEM	0.7828	75	0.01044	5.93	0
MODEL*SECURITY	2.9407	1335	0.0022	1.25	0
SYSTEM*SECURITY	4.6019	445	0.01034	5.87	0
Error	11.7542	6675	0.00176		
Total	27.9378	8639			

Constrained (Type III) sums of squares.

Figure 29. MATLAB multi-way ANOVA table.

The harmonic mean (F-Score) was selected as the ANOVA response variable to capture differences in both the recall and precision performance attributes of the classification models. The ANOVA table demonstrates that each of the corresponding six formal null hypotheses stated in chapter 4 were rejected (Prob>F) with an alpha of 0.05 (95% confidence), definitively suggesting differences in performance does exist among the classification models as measured by the macro harmonic mean. The ANOVA table furthermore confirms interaction among the three identified factors meaning multiple factors should be considered when forming a trading strategy.

Figure 30 shows each of the three factors, namely the classification models, trading systems, and financial securities from left to right, respectively. The variance in mean F-Score can be seen within the graph of each factor (main effect). If a specific factor was to show no difference in mean F-Score, the blue line within each factor would have been more horizontally flat.

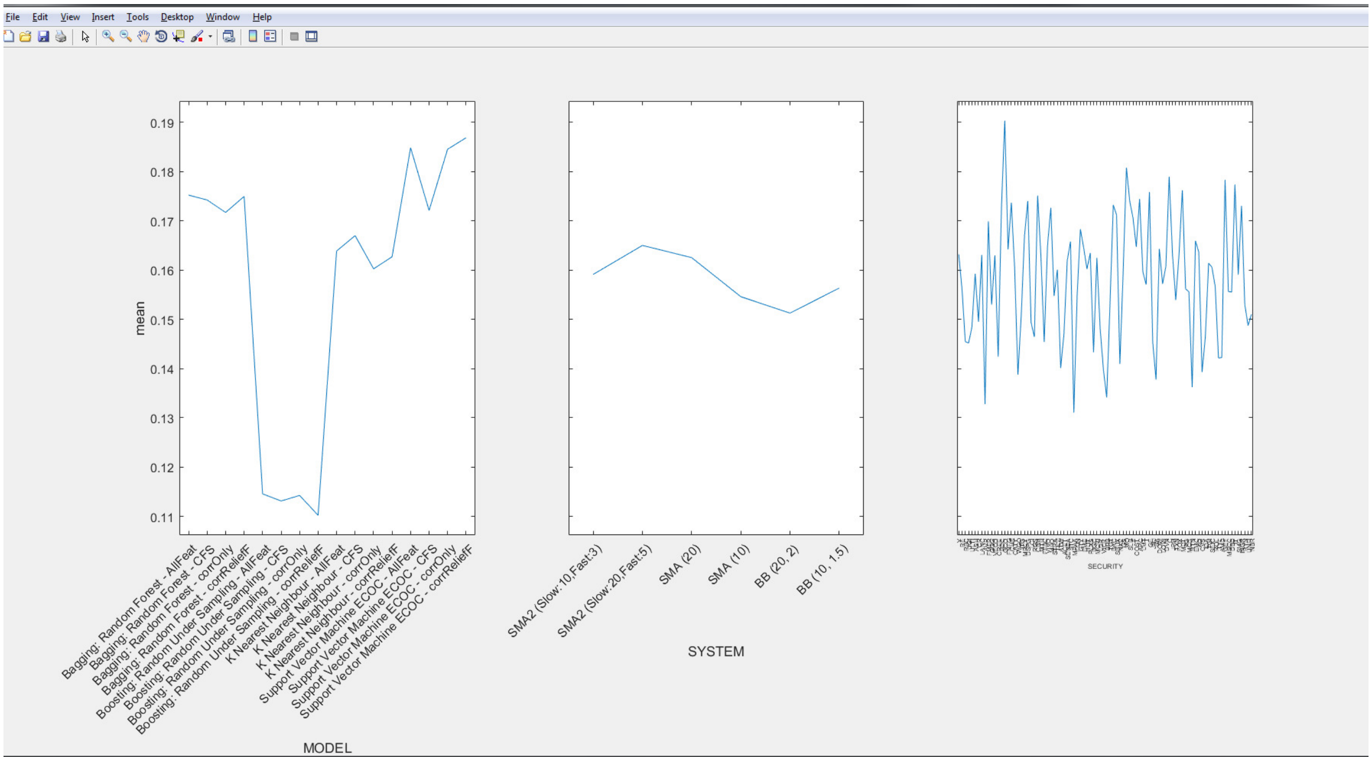


Figure 30. MATLAB Main Effects Plot

The graph representing 16 different classification models on the left shows boosting to have had a significantly different average F-Score than the other model types considered. The graph in the middle representing trading systems shows there was a difference in the model performance based on the trading system selected, suggesting a difference in performance between the less sensitive trend-following systems (SMA and SMA2) and the counter trend systems (BB). The graph on the right of *Figure 30* shows that the performance of the classification model varied with financial security. *Figure 31* through *Figure 33* provided in the subsections below show the difference in F-score among these three significant factors in greater detail.

Factor – trading systems

Figure 31 shows the difference in average F-score among the trading systems considered. The figure highlights the SMA2(20,5) trading system had an average F-Score statistically greater than other trading systems. The trading system BB(20,2) had the lowest average macro F-Score among the trading systems considered.

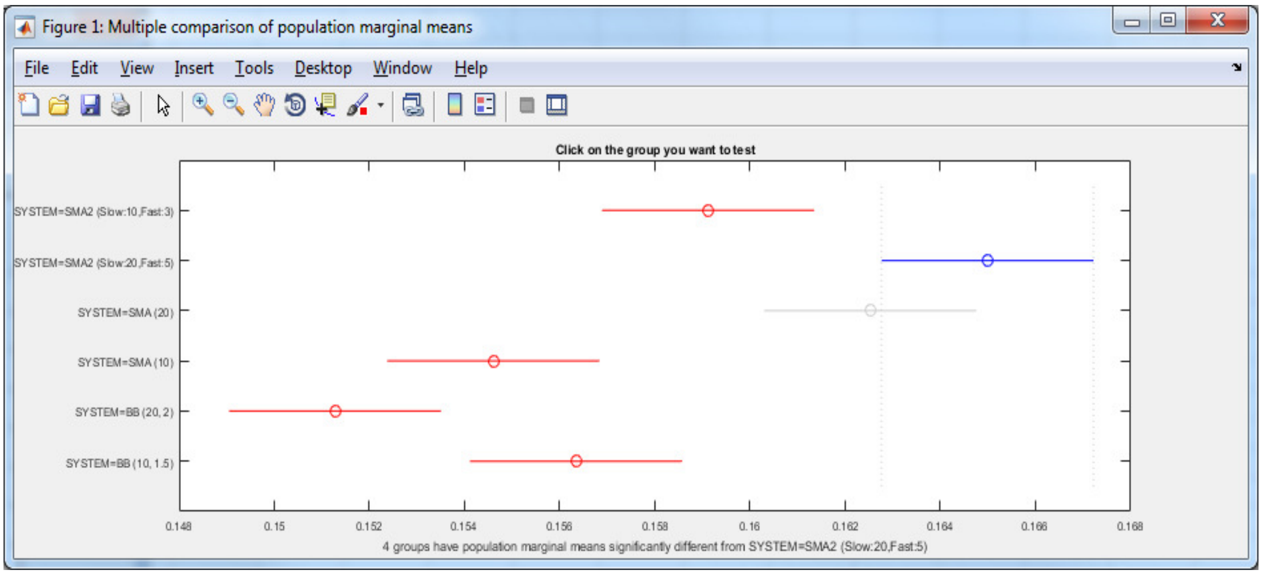


Figure 31. MATLAB comparison among trading systems.

Factor – model

Figure 32 shows the difference in F-score among the 16 classification models considered (four model types each trained with four different feature selection approaches). The figure highlights the SVM models had an F-Score statistically greater than other classification models considered. The bagging ensembles were also statistically different than boosting and most KNN ensemble types. The mean F-Score shows that different model types had a different balance between precision and recall, which implies different performance characteristics, but not necessarily the greatest improvement beyond random chance.

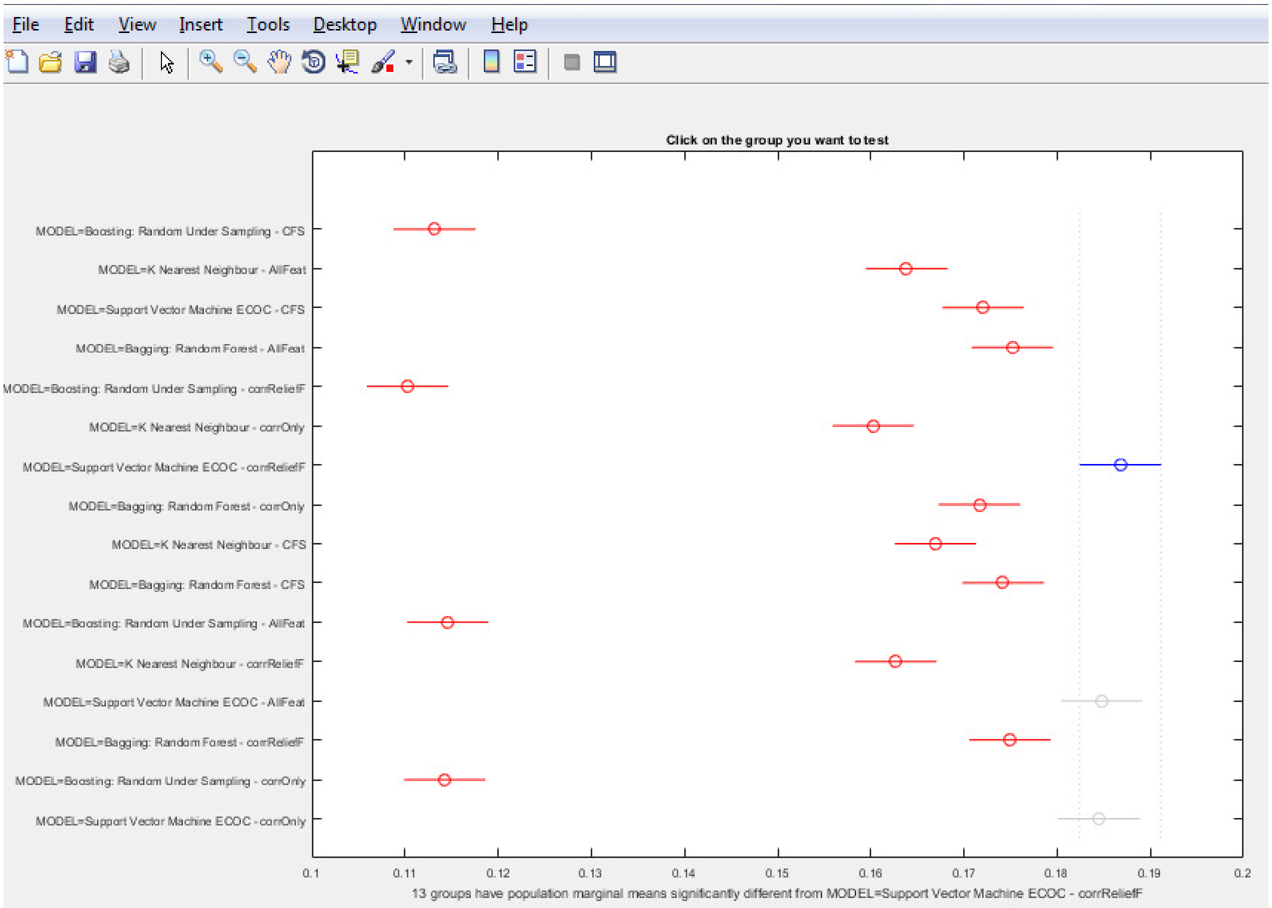


Figure 32. MATLAB comparison among models.

Factor – financial security

Figure 33 shows the difference in average F-score among the 90 financial securities considered. The figure shows that selecting different financial securities resulted in different performance of the classification model as measured by the F-Score.

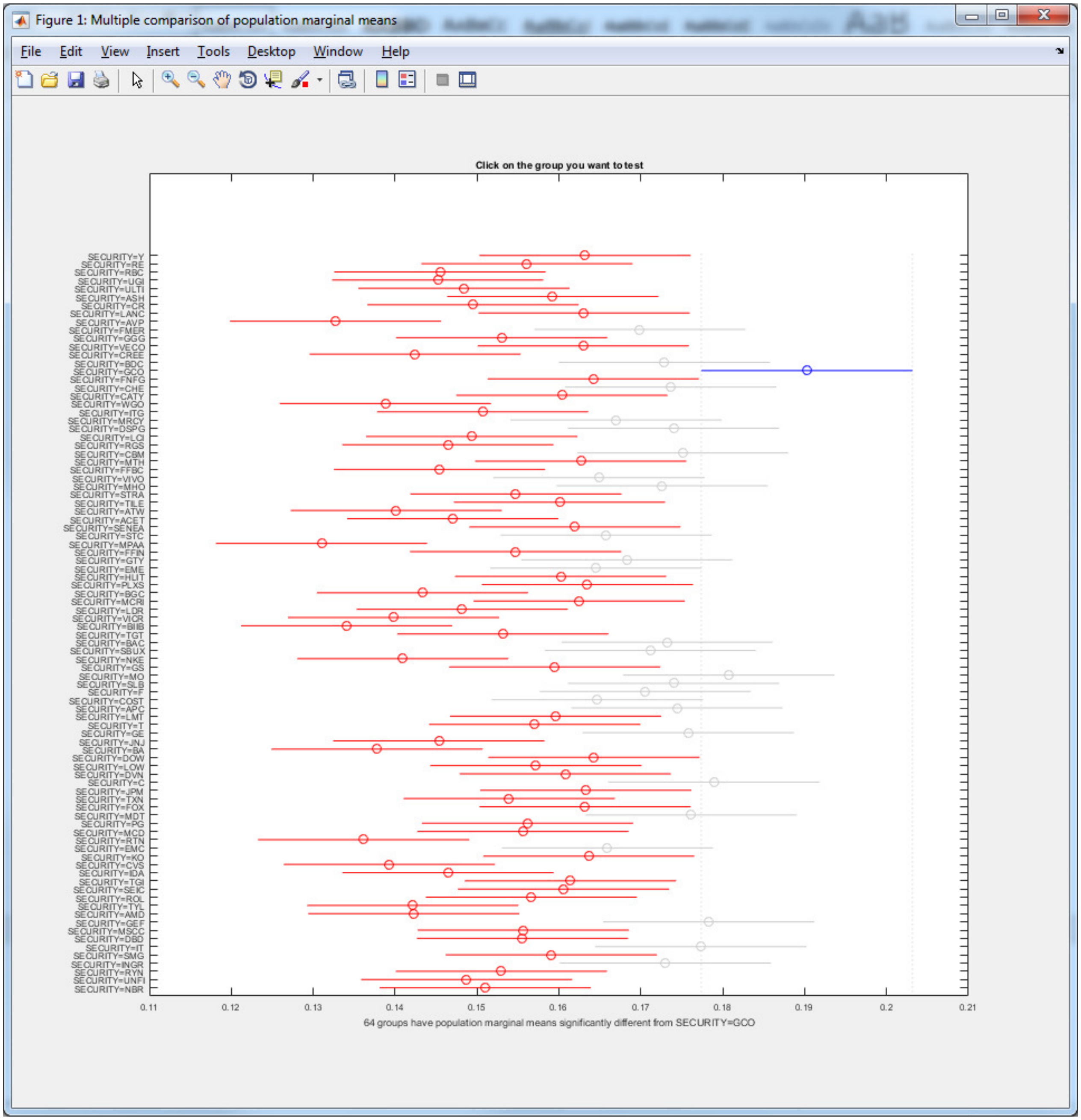


Figure 33. MATLAB comparison among financial securities.

Interaction among factors

Figure 34 shows the interaction between the three factors. All permutations of interactions between the classification model, trading system, and financial security were significant indicating a technical trader should select the trading system, classification model, and securities to be monitored with consideration for each of the other factors. The selection and interaction between the trading system and classification model are under the control of technical trader. The controllable trading system and classification model factors are discussed further in later chapters of this research. The attributes of the financial time series were not the focus in this research and therefore not investigated to determine (or attempt to identify) common time series patterns that lead to greater performance in the classification of an event.

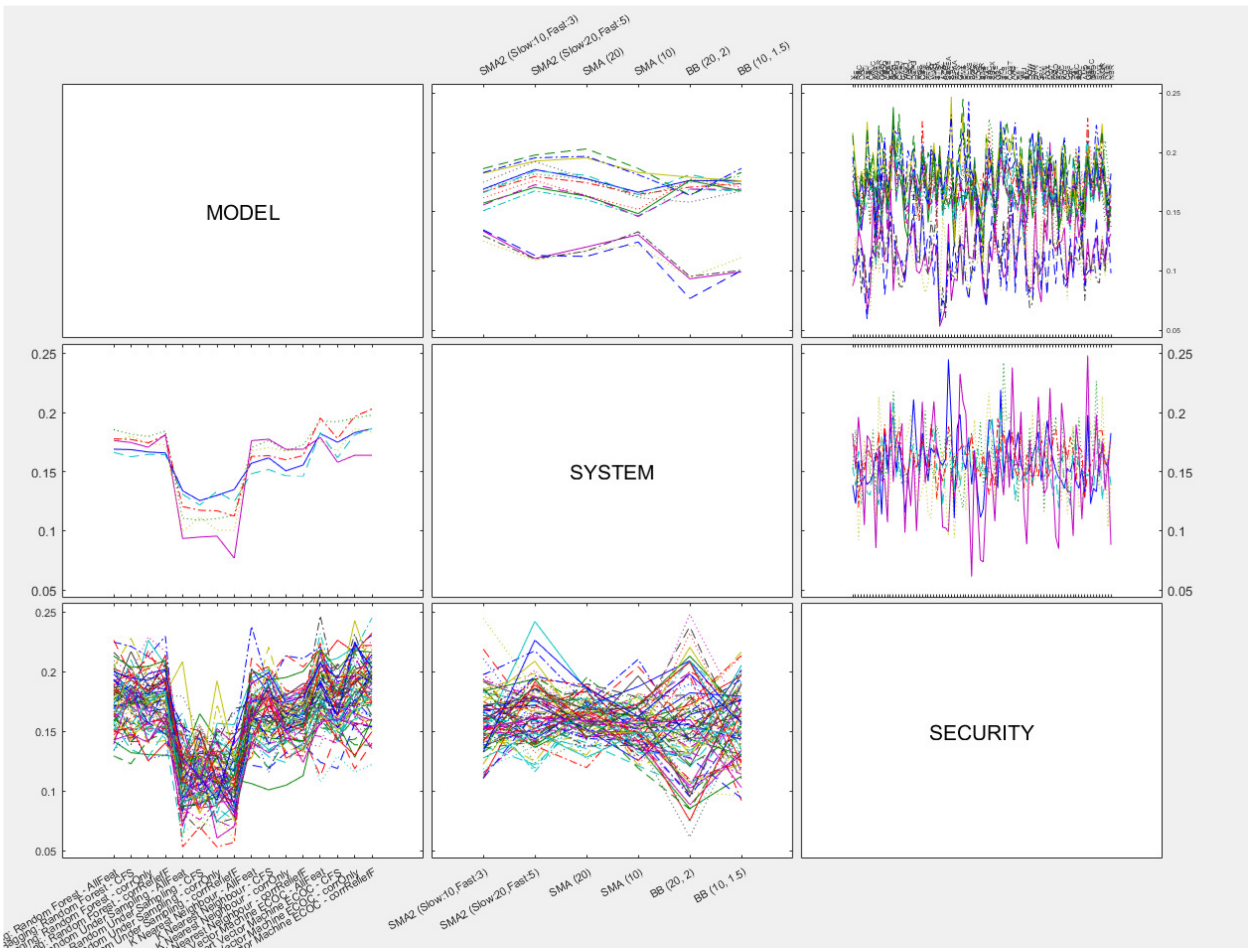


Figure 34: MATLAB Interaction Plot

Conclusion: There was a difference in the performance among the selected ensemble classification models in terms of average F-Score. The differences in performance characteristics was influenced by the technical trading system used to generate the events as well as the underlying financial security.

Model performance compared to random chance

To determine if one specific model in combination with a specific feature selection approach performed statistically better than random chance, a right tail *t*-test using the kappa statistic was employed. Furthermore, a second right tail *t*-test using the MCC was also employed to check the conclusions.

The kappa statistic quantifies the degree of alignment between the predicted event outcome labels and the actual event outcome labels while accounting for the performance achievable by random chance. A kappa statistic of 1.0 implies perfect alignment between the models' predicted labels and the actual class labels; a kappa statistic that is negative suggests the classifier performs worse that random chance, or more simply, worse than nothing. Therefore, to support the statement that a classifier can perform better than random chance, the kappa value had to be statistically larger than zero, which would indicate the classification model is at least statistically better than random chance ($\kappa > 0$).

The MCC quantifies the correlation of the predicted event outcome labels with the actual event outcome labels. An MCC greater than zero implies a positive correlation, a negative correlation suggests the classification model provides misdirection (worse than nothing), and an MCC equal to zero implies there is no correlation between the predicted outcome labels and the actual event outcome labels. Therefore, to support the statement that a classifier can perform better than random chance, the MCC value had to be statistically greater than zero.

The following is the standard equation for a student *t*-test, where $\bar{x}$ is the sample mean, μ is the hypothesized population mean, s is the sample standard deviation, and *n* is the sample size. Under the null hypothesis, the test statistic assumes a student's *t* distribution with *n* – 1 degrees of freedom.

$$t = \frac{\bar{x} - \mu}{s/\sqrt{n}}$$

The *t*-test alternative hypothesis (H_a) is that the mean kappa and MCC values are greater than zero. Therefore, a value of 1 (or true) indicates the rejection of the null hypothesis (H_0) that the kappa or MCC value are less than or equal to zero at the Alpha significance level 0.05.

All classification models had a kappa and MCC value that was statistically greater than zero except Boosting: Random Under Sampling in combination with Extreme Multicollinearity Removed + Relief-F as a feature selection approach. The designated top-performing classification model within each ensemble type is highlighted in

Table 14 based on kappa value results and with consideration for the MCC results. The highlighted rows in Table 14 identify the most favourable feature selection approach for each classification model type. Table values are the average of all trading systems, all securities, and all models, sliced by Model. Table 15 provides a summary of the MCC results.

Table 14

Kappa Value for Classification Models

Classification Model	Ave. Kappa	St Dev. Kappa	Max. Kappa	Significantly > 0
Bagging: Random Forest All Features	0.044	0.093	0.495	1
Bagging: Random Forest Correlation based Feature Selection	0.021	0.105	0.536	1
Bagging: Random Forest Extreme Multicollinearity Removed	0.038	0.095	0.387	1
Bagging: Random Forest Extreme Multicollinearity Removed + Relief F	0.037	0.100	0.594	1
Boosting: Random Under Sampling All Features	0.015	0.071	0.306	1
Boosting: Random Under Sampling Correlation based Feature Selection	0.010	0.071	0.268	1
Boosting: Random Under Sampling Extreme Multicollinearity Removed	0.013	0.068	0.306	1
Boosting: Random Under Sampling Extreme Multicollinearity Removed + Relief-F	0.006	0.071	0.248	1
K-Nearest Neighbour All Features	0.022	0.083	0.495	1
K-Nearest Neighbour Correlation-based Feature Selection	0.010	0.094	0.339	1
K-Nearest Neighbour Extreme Multicollinearity Removed	0.017	0.071	0.361	1
K-Nearest Neighbour Extreme Multicollinearity Removed + Relief-F	0.020	0.079	0.322	1
Support Vector Machine All Features	0.030	0.104	0.409	1
Support Vector Machine ECOC Correlation-based Feature Selection	0.017	0.094	0.346	1
Support Vector Machine ECOC Extreme Multicollinearity Removed	0.029	0.103	0.404	1
Support Vector Machine ECOC Extreme Multicollinearity Removed + Relief-F	0.030	0.106	0.370	1

Table 15

MCC Values for Classification Models

Classification Model	Ave. MCC	StDev. MCC	Max. MCC	Significantly > 0
Bagging: Random Forest All Features	0.059	0.117	0.553	1
Bagging: Random Forest Correlation based Feature Selection	0.025	0.124	0.575	1
Bagging: Random Forest Extreme Multicollinearity Removed	0.051	0.122	0.423	1
Bagging: Random Forest Extreme Multicollinearity Removed + Relief F	0.051	0.124	0.619	1
Boosting: Random Under Sampling All Features	0.016	0.085	0.366	1
Boosting: Random Under Sampling Correlation based Feature Selection	0.011	0.086	0.367	1
Boosting: Random Under Sampling Extreme Multicollinearity Removed	0.014	0.080	0.366	1
Boosting: Random Under Sampling Extreme Multicollinearity Removed + Relief-F	0.005	0.084	0.266	0
K-Nearest Neighbour All Features	0.034	0.110	0.553	1
K-Nearest Neighbour Correlation based Feature Selection	0.017	0.116	0.384	1
K-Nearest Neighbour Extreme Multicollinearity Removed	0.030	0.097	0.378	1
zyK-Nearest Neighbour Extreme Multicollinearity Removed + Relief-F	0.032	0.109	0.381	1
Support Vector Machine All Features	0.031	0.114	0.455	1
Support Vector Machine ECOC Correlation-based Feature Selection	0.019	0.106	0.373	1
Support Vector Machine ECOC Extreme Multicollinearity Removed	0.031	0.112	0.442	1
Support Vector Machine ECOC Extreme Multicollinearity Removed + Relief-F	0.031	0.116	0.390	1

Landis and Koch (1977) proposed the following as standards for strength of agreement for the kappa coefficient: ≤0.00=poor, 0.01–0.20=slight, 0.21–0.40=fair, 0.41–0.60=moderate, 0.61–0.80=substantial, and 0.81–1.00=almost perfect. The average kappa value across all profiles in the verification set was 0.023 suggesting slight agreement. The model with the greatest average kappa statistic was Bagging: Random Forest All Features, which had an average kappa value of 0.044 across all trading systems and financial securities. The most favourable SVM benchmark model had an average kappa statistic 0.03 across all trading systems and financial securities. The model with the greatest average MCC across all trading systems and securities was also Bagging: Random Forest All Features, which had an average MCC of 0.059 suggesting minor alignment.

Conclusion: It is possible to develop an ensemble classification model that is able to predict the outcome of an event using a diverse set of technical market indicators with a performance level that can be considered superior to random chance. Although performing better than random chance (kappa and MCC > 0), the degree of competitive advantage offered by the classification models is best described as slight.

Direct model comparisons

As all three factors were shown to be significant in the performance of the classification models, any direct comparison between the models needed to consider both the financial security and technical trading system. A paired-sample *t*-test using the kappa statistic was employed to address the interaction among the factors. Table 16 presents a summary of the abbreviations used as row and column headers in the comparison matrix for kappa and MCC paired comparisons shown in Table 17 and Table 18, respectively.

Table 16

Legend for Model Comparison Tables

Abbreviation	Ensemble Method	Specific Algorithm (if Applicable)	Classifier	Feature Selection Approach
Bag: RF-T: All	Bagging	Random Forest	Tree	All Features
Bag: RF-T: CFS	Bagging	Random Forest	Tree	Correlation Feature Selection
Bag: RF-T: Corr	Bagging	Random Forest	Tree	Multicollinearity Removed
Bag: RF-T: Rlf-F	Bagging	Random Forest	Tree	Multicollinearity Removed + Relief-F
Boost: RUS-T: All	Boosting	Random Under Sampling	Tree	All Features
Boost: RUS-T: CFS	Boosting	Random Under Sampling	Tree	Correlation Feature Selection
Boost: RUS-T: Corr	Boosting	Random Under Sampling	Tree	Multicollinearity Removed
Boost: RUS-T: Rlf-F	Boosting	Random Under Sampling	Tree	Multicollinearity Removed + Relief-F
Sub: KNN: All	Subspace	N/A	K-Nearest Neighbour	All Features
Sub: KNN: CFS	Subspace	N/A	K-Nearest Neighbour	Correlation Feature Selection
Sub: KNN: Corr	Subspace	N/A	K-Nearest Neighbour	Multicollinearity Removed
Sub: KNN: Rlf-F	Subspace	N/A	K-Nearest Neighbour	Multicollinearity Removed + Relief-F
SVM: ECOC: All	N/A	Error Correcting Codes	Support Vector Machines	All Features
SVM: ECOC: CFS	N/A	Error Correcting Codes	Support Vector Machines	Correlation Feature Selection
SVM: ECOC: Corr	N/A	Error Correcting Codes	Support Vector Machines	Multicollinearity Removed
SVM: ECOC: Rlf-F	N/A	Error Correcting Codes	Support Vector Machines	Multicollinearity Removed + Relief-F

Model comparison kappa and MCC tables

Table 17

Kappa Statistic Paired Comparison

ROW > (Better Than) COLUMN	Bag: RF-T: All	Bag: RF-T: CFS	Bag: RF-T: Corr	Bag: RF-T: Rlf-F	Boost: RUS-T: All	Boost: RUS-T: CFS	Boost: RUS-T: Corr	Boost: RUS-T: Rlf-F	Sub: KNN: All	Sub: KNN: CFS	Sub: KNN: Corr	Sub: KNN: Rlf-F	SVM: ECOC: All	SVM: ECOC: CFS	SVM: ECOC: Corr	SVM: ECOC: Rlf-F
Bag: RF-T: All	N/A	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Bag: RF-T: CFS	0	N/A	0	0	0	1	0	1	0	1	0	0	0	0	0	0
Bag: RF-T: Corr	0	1	N/A	0	1	1	1	1	1	1	1	1	0	1	1	0
Bag: RF-T: Rlf-F	0	1	0	N/A	1	1	1	1	1	1	1	1	0	1	0	0
Boost: RUS-T: All	0	0	0	0	N/A	0	0	1	0	0	0	0	0	0	0	0
Boost: RUS-T: CFS	0	0	0	0	0	N/A	0	0	0	0	0	0	0	0	0	0
Boost: RUS-T: Corr	0	0	0	0	0	0	N/A	1	0	0	0	0	0	0	0	0
Boost: RUS-T: Rlf-F	0	0	0	0	0	0	0	N/A	0	0	0	0	0	0	0	0
Sub: KNN: All	0	0	0	0	0	1	1	1	N/A	1	0	0	0	0	0	0
Sub: KNN: CFS	0	0	0	0	0	0	0	0	0	N/A	0	0	0	0	0	0
Sub: KNN: Corr	0	0	0	0	0	1	0	1	0	0	N/A	0	0	0	0	0
Sub: KNN: Rlf-F	0	0	0	0	0	1	0	1	0	1	0	N/A	0	0	0	0
SVM: ECOC: All	0	0	0	0	1	1	1	1	1	1	1	1	N/A	1	0	0
SVM: ECOC: CFS	0	0	0	0	0	0	0	1	0	0	0	0	0	N/A	0	0
SVM: ECOC: Corr	0	0	0	0	1	1	1	1	1	1	1	1	0	1	N/A	0
SVM: ECOC: Rlf-F	0	0	0	0	1	1	1	1	1	1	1	1	0	1	0	N/A

Table 18

Mathews Correlation Coefficient (MCC) Paired Comparison

ROW > (Better Than) COLUMN	Bag: RF-T: All	Bag: RF-T: CFS	Bag: RF-T: Corr	Bag: RF-T: Rlf-F	Boost: RUS-T: All	Boost: RUS-T: CFS	Boost: RUS-T: Corr	Boost: RUS-T: Rlf-F	Sub: KNN: All	Sub: KNN: CFS	Sub: KNN: Corr	Sub: KNN: Rlf-F	SVM: ECOC: All	SVM: ECOC: CFS	SVM: ECOC: Corr	SVM: ECOC: Rlf-F
Bag: RF-T: All	N/A	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Bag: RF-T: CFS	0	N/A	0	0	1	1	1	1	0	1	0	0	0	0	0	0
Bag: RF-T: Corr	0	1	N/A	0	1	1	1	1	1	1	1	1	1	1	1	1
Bag: RF-T: Rlf-F	0	1	0	N/A	1	1	1	1	1	1	1	1	1	1	1	1
Boost: RUS-T: All	0	0	0	0	N/A	0	0	1	0	0	0	0	0	0	0	0
Boost: RUS-T: CFS	0	0	0	0	0	N/A	0	0	0	0	0	0	0	0	0	0
Boost: RUS-T: Corr	0	0	0	0	0	0	N/A	1	0	0	0	0	0	0	0	0
Boost: RUS-T: Rlf-F	0	0	0	0	0	0	0	N/A	0	0	0	0	0	0	0	0
Sub: KNN: All	0	0	0	0	1	1	1	1	N/A	1	0	0	0	1	0	0
Sub: KNN: CFS	0	0	0	0	0	0	0	1	0	N/A	0	0	0	0	0	0
Sub: KNN: Corr	0	0	0	0	1	1	1	1	0	1	N/A	0	0	1	0	0
Sub: KNN: Rlf-F	0	0	0	0	1	1	1	1	0	1	0	N/A	0	1	0	0
SVM: ECOC: All	0	0	0	0	1	1	1	1	0	1	0	0	N/A	1	0	0
SVM: ECOC: CFS	0	0	0	0	0	0	0	1	0	0	0	0	0	N/A	0	0
SVM: ECOC: Corr	0	0	0	0	1	1	1	1	0	1	0	0	0	1	N/A	0
SVM: ECOC: Rlf-F	0	0	0	0	1	1	1	1	0	1	0	0	0	1	0	N/A

In the context of this research, each classification model provided an outcome label for each event, which is a categorical rating. Each event was also assigned an actual event outcome label (categorical rating) based on proximity to existing cluster centres established with the training data. The kappa statistic measured the degree agreement between the actual event outcome label and the classification model event outcome label. The MCC measured the degree of correlation between the two sets of labels. Table 17 and Table 18 evaluate if the row model had a statistically more favourable kappa and MCC metric than the column model (row > column). This was investigated using a right-tail pairwise t-test. All 16 models were tabled in the matrix. If the row, column intersection in the table show a value of 1, then the t-test rejected the null hypothesis that the two models have the same performance in favor of the alternate hypothesis that the row model population had a mean greater than the column model (row > column) at an alpha of 0.05.

Conclusion: Some classification models performed statistically superior in comparison to the other models based on the kappa and MCC as measures of alignment between the predicted categorical label (event outcome) and the actual categorical rating. The bagging random forest ensemble model using all available candidate features performed statistically greater than the other classification models considered, including the benchmark SVM classification models based on the kappa and MCC values.

Investigation of frequently selected candidate features

A total of 1,597 candidate features were calculated for each event. The feature selection method of all features with multi-collinearity removed averaged 644 candidate features for each event. The feature selection method of relief-F after pre-screening for multicollinearity averaged to 187 candidate features per event, and the correlation feature selection approach consistently resulted in the least number of features included in the ensemble with an average of 19 per event as shown in Table 19.

Table 19

Average Number of Features Selected per Approach

Feature Selection Approach	Average Number of Features in Feature Set
All Features	1,597 (constant)
Correlation Feature Selection	19
Multicollinearity Removed	644
Multicollinearity Removed + Relief-F	187

Frequently selected candidate features

Each profile consisted of a financial security and a trading system. The resulting total based on 90 securities and six trading systems was 540 profiles in the verification set. Each classification model had four feature selection permutations. Some features were selected more often by the feature selection approaches. Table 20 shows the top 20 candidate features selected across the 540 profiles considered in the verification set. The purpose of this table is to identify which of the technical indicators representing trend, momentum, volatility and volume were most commonly selected.

Table 20

Top 20 selected Candidate Features

Indicator	Time Dimension	Time View Point	Derivative
Efficiency Ratio	5	1	Obs
A/D oscillator	42	21	Acc
Linear Regression Slope	63	10	Acc
Directional Movement	15	1	Obs
Directional Movement	5	1	Obs

Indicator	Time Dimension	Time View Point	Derivative
Efficiency Ratio	42	2	Acc
Monitoring Duration Count	N/A	N/A	N/A
A/D oscillator	21	10	Acc
A/D oscillator	63	21	Acc
A/D oscillator	32	21	Acc
Directional Movement	5	2	Obs
A/D oscillator	32	15	Acc
Linear Regression Slope	42	5	Acc
Efficiency Ratio	10	2	Acc
Linear Regression Slope	63	5	Acc
A/D oscillator	42	15	Acc
A/D oscillator	21	5	Acc
Linear Regression Slope	63	3	Acc
A/D oscillator	63	10	Acc
Efficiency Ratio	42	3	Acc

Note. Acc = Acceleration; Obs = Observation.

The most commonly selected candidate features were the linear regression slope and directional movement representing the trend, the accumulation/distribution (A/D) oscillator representing momentum, and the efficiency ratio representing volatility. Interestingly, candidate features representing the acceleration (speed) in which longer time dimensioned trend and momentum indicators changed were selected most frequently across the different feature selection approaches. One interpretation of this result is the rate of change of candidate features that are less sensitive to market noise provided the most insight of market sentiment.

CHAPTER VI: DISCUSSION

Significance of Results

Technical trading imposes discipline on an individual's trading practices but requires commitment to following the trading signals generated by the technical trading system. This research did not evaluate if technical trading provides an advantage over a buy and hold strategy. Instead, this research investigated if traders who do employ a technical trading strategy are able to gain an advantage by evaluating trading opportunities as events, using a variety of measures of market sentiment within a classification model to anticipate the outcome of an event.

This research used a variety of technical indicators derived from financial time series data to measure various aspects of market sentiment as well as account for the characteristics of time series data. All, or a subset, of the technical indicators available as candidate features were then selected and used as predictors within ensemble models to classify the outcome of an event defined by a technical trading system. The intent was to use the assigned outcome classification label as information to screen opportunities and filter out the common majority of events that were anticipated to have an undesirable or less than average expected return. If a technical trader is able to screen out the majority of insignificant events, the trader would avoid transaction costs and potentially increase returns through more efficient capital allocation.

The findings of this research are complementary to the findings of R. Dash and Dash (2016), who concluded it is more profitable to make trading decisions using combinations of technical indicators with computational intelligence tools than to use any one particular technical indicator as a decision system. This research suggested is it possible for a technical trader to use a variety of technical indicators in combination with ensemble models to gain additional information from financial time series pricing data. This research demonstrated that ensemble models using technical market indicators could provide a slight to marginal advantage beyond random chance based on the average kappa values observed.

Response Mode Selection: Exploratory Data Sets

Four exploratory data sets were used to investigate the impact of the response mode, with each data set reflecting a method used to group the outcomes of events into discrete outcome classification labels. The results of the exploratory data sets suggest there are considerable differences in the performance of the classification models based on how the events were grouped together into outcome classes.

Table 21 shows the key performance measures used in the research associated with each of the four response modes. Although a specific profile within a particular response mode could have a more favourable or less favourable result, the differences in the overall averages provide an indication of which response mode was likely to offer the greatest competitive advantage. Table 21 also provides the classification performance measures as an average across all the models associated with the response mode scenario. The 'reference' value provides a baseline value for perspective to account for the number of groupings contained within each response mode. For example, if clustering events into 6 groupings, there is a 1 in 6 random chance of selecting correct, in binary there is a 1 in 2 random chance of selecting correct. Based on Table 21, the response mode that clustered event outcomes into six classes (cluster multi-class) provided the most favourable overall kappa value and favourable recall and precision metrics relative to random chance, shown as the 'reference' point.

Table 21

Key Performance Measures Across Response Modes

Performance Metric	Cluster Multi-class (6)	Binning Multi-class (4)	Cluster Binary (2)	Bin Binary (2)
Reference	*1/6 = 0.167*	*1/4 = 0.250*	*1/2 = 0.500*	*1/2 = 0.500*
Macro F-Score	0.159	0.217	0.347	0.404
Macro Recall	0.198	0.267	0.396	0.481
Macro Precision	0.220	0.297	0.445	0.432
Overall Accuracy	0.385	0.468	0.707	0.562
Kappa	0.023	0.009	0.010	0.001
MCC	0.029	0.010	0.011	0.002

Note. MCC = Mathews Correlation Coefficient.

The verification data set was used to explore the research questions set out within this research work. As discussed in Chapter 4, the verification data set consisted of 90 financial securities independently selected from the 30 financial securities used in the exploratory data sets. This experimental design decision provided a degree of external validation by expanding the number of financial securities evaluated when exploring the hypotheses of this research work.

Observations and Interpretations among Exploratory Data Sets

Response mode: Cluster versus binning

The response mode refers to the method used to group events together into broad classification labels within the training data. Two response modes were investigated to group the outcomes of events into classification labels, namely clustering and binning.

When the response mode was set to clustering, the application grouped each historical event outcome into six mutually exclusive outcome classes using the K-means clustering algorithm based on two dimensions, namely the return and active duration of the event. The resulting six clusters were named terrible, undesirable, unfavourable, favourable, desirable, and excellent, with the cluster labelled terrible having the least favourable average return among events contained within the cluster and excellent having the most favourable average return among events contained in the cluster, respectively.

When the response mode was set to binning, the application grouped each historical event into four mutually exclusive classes based on the return of the event. The resulting four bins were labelled terrible, unfavourable, favourable, and excellent, with the bin labelled terrible containing the events with the least favourable event returns and the bin labelled excellent having the most favourable event returns, respectively.

The “clustering” response mode yielded better performance from the classification models than the “binning” response mode. The binning algorithm only considered the event return and did not factor in the duration of the event. This research work argues that clustering performed more favourably over binning as the k-means clustering algorithm was able to better group similar events than binning by giving consideration to both the return of the event and duration of the event.

A positive correlation between the event duration and the event return was demonstrated, provided as *Figure 35*. Specifically, the event return and the duration are positively correlated with rho = 0.52 among all events from all profiles using the Spearman’s rank correlation. The resultant figure shows the relationship with duration (in days) plotted on the X-axis and the event return plotted on the Y-axis across all securities and trading systems considered in the evaluation training data set.

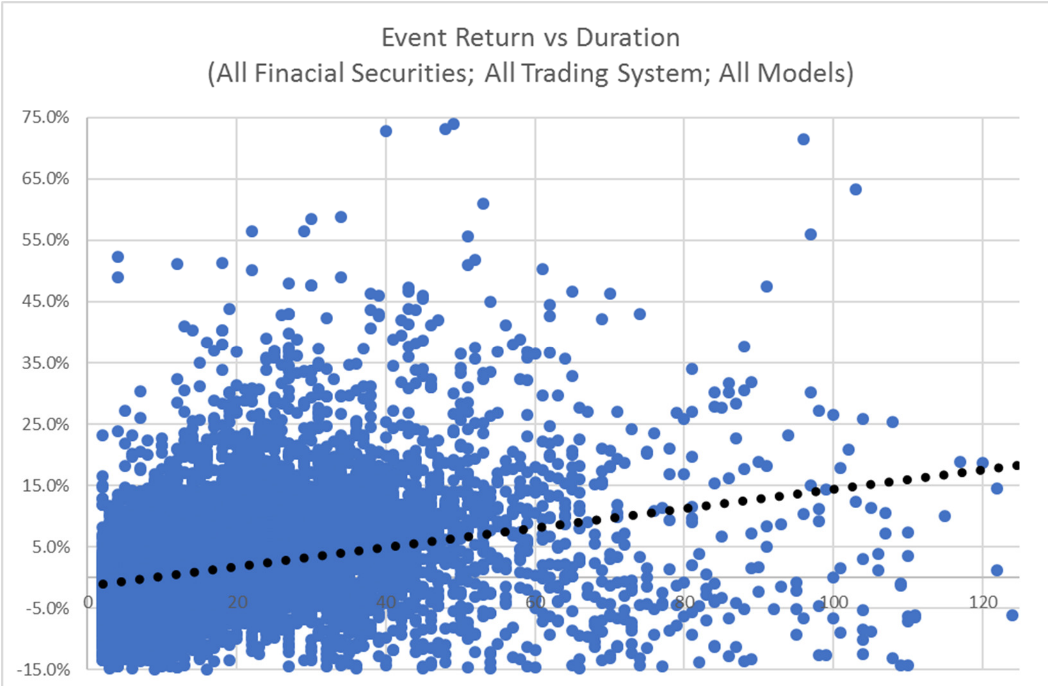


Figure 35. Event return versus duration in days.
Note. Not all events are shown in this graph; the graph scale has been adjusted to remove extreme values.

This research indirectly examined if technical traders are able to classify outcomes of events that represent fundamental shifts in the financial security's price equilibrium. Many events defined by a technical trading system are shorter in duration, have an immaterial negative or positive return, and are generally a result of market volatility in the financial security pricing rather than meaningful shifts in market equilibrium. This research work argues that meaningful price movements are both directional and sustained. By clustering the historic event outcomes with consideration for both the duration and return, the ensemble models were trained using events and the corresponding measures of market behaviour (candidate features) that better differentiate between price volatility as more significant changes in market equilibrium. In contrast, binning of the event returns did not consider the duration of events when forming classification labels.

Response type: Multi-class versus binary

The response type refers to the number of event outcome groupings. There were two classification types considered, namely binary and multi-class. In binary classification, events were grouped into two mutually exclusive classes. In multi-class classification, the event outcome classes reflected different degrees of favourable and unfavourable depending on the response mode used to form the groups.

The multi-class classification of events generally yielded better model performance than binary classification. The researcher inferred that multi-class performed more favourably as the distribution of event returns across the trading systems had a large kurtosis value and very long tails. The distribution of returns among all events generated by the six trading systems is shown in Table 22 and *Figure 36*.

Table 22

Descriptive Statistics of Events

Descriptive Statistics of Events	
Mean	0.55%
Standard Error	0.04%
Median	-0.71%
Mode	0.00%
Standard Deviation	6.27%
Kurtosis	24.21
Skewness	2.52
Minimum	-60.21%
Maximum	101.64%
Event Count (All Profiles)	29,380

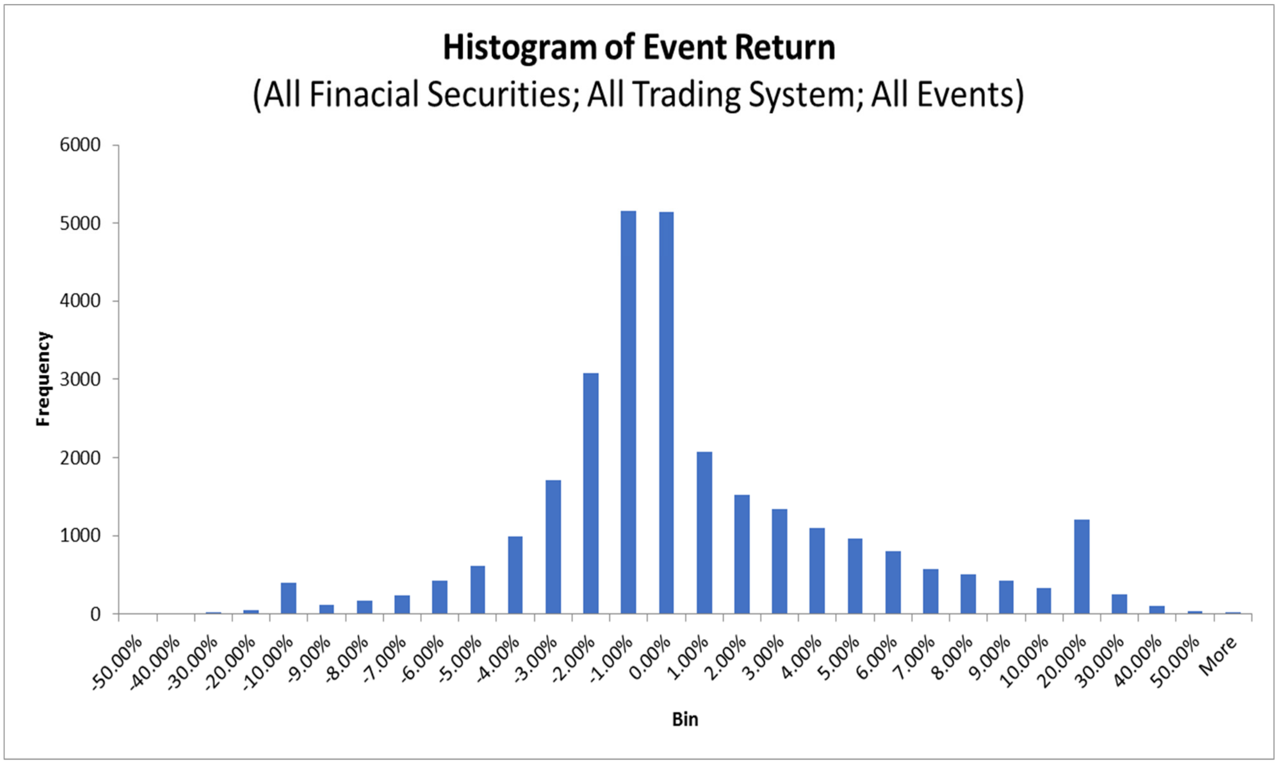


Figure 36. Histogram of event returns.

The verification data set encompassed a total of 29,380 events across the 540 profiles. As many events had a return close to zero (median = -0.71%, kurtosis = 24.21), it is likely that the ensemble models had difficulty differentiating between the positive events and negative events. In the case of binary classification, the events representing significant changes in the market equilibrium were blended and likely lost among the more frequently occurring events that had returns close to zero. For example, in the case of the response variable "binary – binning," an event with a 20% return and an event with a 0.3% return were both assigned the outcome label of "favourable," as both events had a positive return. Correspondingly, an event with a -0.2% return and an event with a -10% return were both assigned an outcome label "unfavourable." In this example, there are likely only minor differences in market sentiment between an event with a – -0.2% return and an event with a +0.3%, but the frequency of the events with insignificant returns is much greater. In the case of multi-classification, events representing significant changes in market equilibrium were further separated from the median value and likely clustered within a differentiating label.

Discussion Around Identified Factors

This research considered three factors as potential influences to the performance of the classification models, namely the financial security, the technical trading system, and type of model with a particular feature selection approach. In this section, each of those factors are independently discussed in the context of classification model performance. The following section then discusses the most favourable model performance among the interacting factors to form a recommended strategy.

Factor A: Financial securities

A total of 30 financial securities were selected at random from each of the S&P SmallCap 600, S&P MidCap 400 and S&P 500 index for a total of 90 financial securities in the verification data set. The attributes of an individual financial security's time series were shown to be a statistically significant factor in the variance of performance among the classification models. Table 23 below shows the three most favourable and the three least favourable financial securities in terms of average classification performance across all trading systems and all classification models considered.

Table 23

Most and Least Favourable Securities

Select Security	Average of Macro F-Score	Average of Macro Recall	Average of Macro Precision	Average of Kappa	Average of MCC
Most Favourable					
BDC	0.173	0.209	0.238	0.078	0.093
GEF	0.178	0.222	0.248	0.093	0.107
SLB	0.174	0.208	0.243	0.067	0.086

Select Security	Average of Macro F-Score	Average of Macro Recall	Average of Macro Precision	Average of Kappa	Average of MCC
Least Favourable					
BIIB	0.134	0.179	0.207	-0.013	-0.012
FFBC	0.145	0.188	0.202	-0.013	-0.011
MCRI	0.162	0.191	0.221	-0.017	-0.018

Note. BDC = Belden Inc. BIIB = Biogen Inc.; FFBC = First Financial Bank Corp; GEF = Greif Bros Corp.; MCC = Mathews Correlation Coefficient; MCRI = Monarch Casino and Resort Inc.; SLB = Schlumberger Limited.

Of the three most favourable and least favourable financial securities, two were large cap, two were mid cap, and two were small cap. Based on these results, although there is a discernable variation between the classification model performance among the different securities, there does not appear to be a market capitalization category that would expect to be better suited for technical trading systems.

Table 24 shows the average classification metrics of the financial security SLB by trading system. The purpose of this table is to show how significantly different the F-score and kappa values varied among the various trading systems for an individual security. A similar table could be developed to show the variance in performance among the different types of classification models.

Table 24

Comparison of Trading Systems

Trading System	Average of Macro F-Score	Average of Macro Recall	Average of Macro Precision	Average of Kappa	Average of MCC
BB (10, 1.5)	0.206	0.238	0.266	0.162	0.198
BB (20, 2)	0.130	0.166	0.189	-0.035	-0.033
SMA (10)	0.133	0.185	0.196	-0.009	-0.010
SMA (20)	0.185	0.214	0.269	0.054	0.092

SMA2 (10, 3)	0.149	0.171	0.227	0.050	0.066
SMA2 (20, 5)	0.242	0.273	0.309	0.179	0.203
Average of All Trading Systems	0.174	0.208	0.243	0.067	0.086

Note. BB = Bollinger Bands; SMA = Simple Moving Average; SMA2 = Two Simple Moving Average Crossover.

Although the financial security is a significant factor, the price movements of any specific financial security are not under the control of the trader. If a technical trader could account for the price movements of a particular security in advance then that trader would have perfect foresight. To reduce the influence of any individual financial security, a larger number of financial securities were selected within the verification data set to diversify and reduce the influence of any specific financial security. The concept of diversification is a well-established best practice in financial investing.

Factor B: Technical trading systems

This research investigated three different technical trading systems, with each trading system exploring two different time dimension parameters. The different time dimension parameters monitored different perspectives of the market.

The SMA and SMA2 are considered trend following trading systems. In contrast, the BB system is considered a countertrend system. A trend following system attempts to provide signals aligned with the direction of the financial security's price movements. In contrast, a countertrend trading system attempts to identify when a financial security is overbought or oversold leading to the anticipation of a reversal in the current trend. Table 25 and Table 26 show the actual number of events and the average actual event returns across all events generated within the 540 profiles, grouped using the multi-class clustering labels.

Table 25

Actual Event Frequency (of Evaluation Set) by Trading System

Trading System	**Actual** Event Count by Outcome Label						Total Events
	Trbl	Undesr	Unfav	Fav	Desr	Excl	
BB (10, 1.5)	41	150	338	708	1,019	598	2,854
BB (20, 2)	38	100	156	335	336	148	1,113
SMA (10)	1,719	3,917	2,175	1,125	384	109	9,429
SMA (20)	1,245	3,041	1,255	589	189	80	6,399
SMA2 (10, 3)	1,300	2,224	1,530	774	341	118	6,287
SMA2 (20, 5)	956	1,103	683	331	160	65	3,298
All Trading Systems	5,299	10,535	6,137	3,862	2,429	1,118	29,380

Note. BB = Bollinger Bands; SMA = Simple Moving Average; SMA2 = Two Simple Moving Average Crossover; Trbl = Terrible; Undesr = Undesirable; Unfav = Unfavourable; Fav = Favourable; Desr = Desirable; Excl = Excellent

Table 26

Actual Average Event Return (of Evaluation Set) by Trading System

Trading System	**Actual** Average Event Return by Outcome Label						Average
	Trbl	Undesr	Unfav	Fav	Desr	Excl	
BB (10, 1.5)	-21.8%	-9.0%	-4.2%	0.5%	3.4%	6.6%	1.4%
BB (20, 2)	-22.8%	-7.8%	-2.7%	4.2%	7.5%	12.7%	3.4%
SMA (10)	-2.7%	-1.4%	0.5%	4.6%	9.6%	18.7%	0.2%
SMA (20)	-3.0%	-1.5%	1.3%	6.9%	14.8%	24.6%	0.3%
SMA2 (10, 3)	-3.8%	-1.6%	0.8%	4.7%	10.1%	18.8%	0.3%
SMA2 (20, 5)	-4.5%	-1.7%	2.5%	8.5%	14.3%	22.8%	0.6%
All Trading Systems	-3.7%	-1.6%	0.6%	4.5%	7.5%	12.1%	0.6%

Note. BB = Bollinger Bands; SMA = Simple Moving Average; SMA2 = Two Simple Moving Average Crossover; Trbl = Terrible; Undesr = Undesirable; Unfav = Unfavourable; Fav = Favourable; Desr = Desirable; Excl = Excellent

As shown in Table 26, the BB countertrend trading system generated events that were the most profitable on average, but the BB systems also generated events with greater losses in unfavourable event outcome classifications and less favourable gains in the rarer positive extreme classes. The SMA trading systems (SMA and SMA2) with a shorter time dimension parameter were more sensitive to market volatility and correspondingly generated comparatively more events than trading systems with longer time dimension parameters, as seen in Table 25.

The ANOVA demonstrated the trading systems were a significant factor in the performance of the classification models. Table 27 shows the average classification performance metrics across all securities and models considered by trading system.

Table 27

Average Classification Metrics by Trading System

Trading System	Average of Macro F-Score	Average of Macro Recall	Average of Macro Precision	**Average of Kappa**	**Average of MCC**
BB (10, 1.5)	0.156	0.195	0.206	**0.001**	0.001
BB (20, 2)	0.151	0.197	0.200	**0.001**	0.001
SMA (10)	0.155	0.192	0.222	**0.024**	0.034
SMA (20)	0.163	0.197	0.231	**0.030**	0.040
SMA2 (10, 3)	0.159	0.196	0.222	**0.036**	0.043
SMA2 (20, 5)	0.165	0.205	0.224	**0.043**	0.052
Average of All Trading Systems	0.158	0.197	0.218	**0.023**	0.029

Note. BB = Bollinger Bands; MCC = Mathews Correlation Coefficient; SMA = Simple Moving Average; SMA2 = Two Simple Moving Average Crossover.

Although the BB systems were more profitable in terms of average event return, the ensemble classification models did not perform well in classifying the price movements of countertrend systems based on the kappa values. In contrast, the slower 20-day SMA trading system and the less sensitive SMA2 trading systems showed greater promise of competitive advantage relative to the other trading systems considered. The trend following systems with longer time dimension parameters were more likely to identify fewer events, but those events were more likely to represent more significant changes in market equilibrium than trading systems with shorter time dimensions that were more influenced by short term volatility.

Factor C: Ensemble models

Bagging

Consistent with the findings of Van den Poel et al. (2016), this research demonstrated that random forest ensemble classification models are the most favourable of the ensemble methods considered. As shown with the pairwise comparisons of Table 17 and Table 18 of Chapter 5, the random forest ensemble method using all available features had a kappa value and MCC statistically greater than other classification models considered. Furthermore, as shown in

Table 14 of Chapter 5, this research demonstrated that random forests are able to classify the outcome of an event better than random chance based on the criteria of having a kappa value statistically greater than zero. Table 28 shows the average key performance measures of the random forest ensemble model in combination with feature selection approaches across all security and trading system combinations contained in the verification set.

Table 28

Bagging: Random Forest Performance Metrics

Random Forest Feature Selection Approach	Average of Macro F-Score	Average of Macro Recall	Average of Macro Precision	Average of Overall Accuracy	**Average of Kappa**	Average of MCC
All Features	0.175	0.211	0.249	0.462	**0.044**	0.059
CFS	0.174	0.205	0.238	0.430	**0.021**	0.025
Multicollinearity Removed	0.172	0.210	0.246	0.464	**0.038**	0.051
Multicollinearity + Relief-F	0.175	0.210	0.247	0.460	**0.037**	0.051
Average of all Feature Selection Approaches	0.174	0.209	0.245	0.454	**0.035**	0.047

Note. CFS = Correlation Feature Selection; MCC = Mathews Correlation Coefficient.

Feature selection did not significantly affect bagged ensembles with the exception of the CFS approach, which performed statistically less favourably than the other feature selection approaches considered. As shown in Table 19 of Chapter 5, CFS generated a feature set containing an average of 19 selected candidate features, which is the least number of features compared to the other feature selection approaches by a substantial amount. Aligning with the supporting theory of random forests, including more candidate features in the selected feature set increases the diversity among the bagged ensemble members, which in turn decreases the impact of noise of any one particular feature. Random forest ensemble models showed the greatest potential to create an effective predictor with minimal risk of overfitting as a result of randomness and the law of large numbers, which provided a greater resilience to outliers and noise.

Random forest ensembles can struggle with imbalanced data. Bagging ensemble methods, including random forests, are generally designed to minimize the overall error rate. Consequently, the random forest ensemble method tends to focus more on the prediction accuracy of the majority class resulting in less accuracy for the minority class. Within this research, the random forest model infrequently classified the outcome of an event as rare cluster classification labels (i.e., excellent or desirable). Correspondingly, random forests often failed to classify the more profitable event outcomes.

The random forest ensembles did perform well at identifying the more common events that were likely to be average or below average (e.g undesirable). Therefore, it would be advantageous for a technical trader to use the random forest ensemble model to avoid common events with outcomes that are the less favourable opposed to trying to identify the extreme profitable events directly using the classification models as a directional guide.

Subspace K-nearest neighbour

The subspace ensemble method using KNN classifiers performed more favourably than boosting, but less favourably than bagging or SVM based on the kappa values. Furthermore, as shown in

Table 14 of Chapter 5, KNN subspace ensembles are able to classify the outcome of an event better than random chance based on the criteria that the kappa value is statistically greater than zero. Table 29 shows the average key performance measures of KNN subspace ensembles in combination with feature selection approaches across all security and system combinations contained in the verification set.

Similar to random forest ensembles, within the KNN subspace ensembles CFS was the only feature selection approach that performed statistically different among the feature selection approaches considered. Based on the kappa statistic, the ensemble model using all features performed the best among the KNN subspace models, closely followed by the feature selection approach with multicollinearity removed in combination with relief-F.

Table 29

Subspace: KNN Performance Metrics

Subspace KNN Feature Selection Approach	Average of Macro F-Score	Average of Macro Recall	Average of Macro Precision	Average of overall Accuracy	**Average of Kappa**	Average of MCC
All Features	0.164	0.205	0.240	0.471	**0.022**	0.034
CFS	0.167	0.203	0.235	0.451	**0.010**	0.017
Multicollinearity Removed	0.160	0.204	0.240	0.475	**0.017**	0.030
Multicollinearity + Relief-F	0.163	0.204	0.240	0.473	**0.020**	0.032
Average of all feature selection approaches	0.163	0.204	0.239	0.468	**0.017**	0.028

Note. CFS = Correlation Feature Selection; MCC = Mathews Correlation Coefficient.

It was anticipated that multicollinearity removed in combination with relief-F would be the top performing feature selection approach among KNN subspace ensemble models. The candidate features are derived from time series pricing data of a particular financial security, which leads to higher levels of correlation, and KNN performance can be hindered with large numbers of correlated factors. Instead, the subspace ensemble model considering all candidate features did perform the most favourably, which provided some indirect support to the notion that each feature represents different aspects of market sentiment and brings some value to the model as a collective whole.

One of the fundamental limitations of KNN in the context of this research is the use of related technical indicators as features to define the nearest neighbour. KNN uses a distance metric to find separation between event outcome classes. If many features are included in the KNN classifier, and those features have a high degree of correlation, then the distance between outcome classes becomes smaller, making it more difficult to differentiate between the classes.

In addition, KNN is generally more effective when there are a large number of events available for training. The number of training events generated by each trading system varied considerably, with the more active trading systems such as SMA(10) generating approximately 244 training events on average, while less active trading systems such as BB (20,2) generated only approximately 29 training events on average. As the numbers of training events were small relative to the number of candidate features, the researcher believes the distance metric had difficulty differentiating between classes, leading to less favourable performance than some of the other classification approaches considered.

Boosting with random under sampling

The boosting ensemble using a random under sampling method performed least favourably among the classification models considered. The RUSBoost classification ensembles collectively had an average kappa statistically significantly smaller than the other classification models investigated. Furthermore, the competitive advantage beyond random chance offered by boosting ensembles is only slightly greater than zero.

Table 30 shows the average key performance measures of boosting random under sampling ensembles in combination with the feature selection approaches across all profiles (security and system combinations) contained in the verification set. Similar to the other ensemble models investigated, using all available candidate features provided the greatest kappa value for boosting, followed by the feature selection approach that removed multicollinearity. The boosting model using multicollinearity removed in combination with relief-F as a feature selection approach did have an MCC average value greater than zero although not statistically greater than zero.

Table 30

Boosting: Random Under Sampling Performance Metrics

Average of All Securities All Trading Systems by Feature Selection	Average of Macro F-score	Average of Macro Recall	Average of Macro Precision	Average of Overall Accuracy	**Average of Kappa**	Average of MCC
All Features	0.115	0.170	0.177	0.234	**0.015**	0.016
CFS	0.113	0.170	0.168	0.241	**0.010**	0.011
Multicollinearity Removed	0.114	0.168	0.174	0.236	**0.013**	0.014
Multicollinearity + Relief-F	0.110	0.166	0.167	0.229	**0.006**	0.005
Average of all feature selection approaches	0.113	0.169	0.171	0.235	**0.011**	0.012

Note. CFS = Correlation Feature Selection; MCC = Mathews Correlation Coefficient.

Boosting ensemble methods attempt to improve classification performance by focusing on misclassified training tuples in the sequential development of additional ensemble members, which introduces the risk of overfitting to the training data. The results of this research support the notion that as the noise level grows in the data set, the boosting ensembles decrease in performance in comparison to other ensemble methods, such as bagging, that are more tolerant to noise. Although performing poorly, boosting ensembles were far more likely to predict the rarer event outcome labels in comparison to the other ensemble methods that look to minimize the overall error, although those predictions were often incorrect.

This research argues that increases in performance produced by boosting ensembles are dependent on the particular characteristics of the data set in addition to the features of the classifiers. When the training data contained outlier events, and the ensemble under sampled the majority classes while subsequently overtraining to the rarer training events, the resulting model performance demonstrated to be only slightly better than random chance.

Support vectors machines (ECOC)

Consistent with the work of Tay and Cao (2001), this research demonstrated that it is advantageous to apply SVMs to forecast financial time series. This research suggests the SVM ECOC classification model performance is statistically better than random chance based on the kappa value.

Table 31 shows the average key performance measures of SVMs using ECOC with the feature selection approaches across events from all security and system combinations contained in the verification set.

Table 31

Support Vector Machines: ECOC Performance Metrics

Average of All Securities All Trading Systems by Feature Selection	Average of Macro Fscore	Average of Macro Recall	Average of Macro Precision	Average of Overall Accuracy	**Average of Kappa**	Average of MCC
All Features	0.185	0.208	0.220	0.397	**0.030**	0.031
CFS	0.172	0.202	0.207	0.390	**0.017**	0.019
Multicollinearity Removed	0.184	0.207	0.218	0.399	**0.029**	0.031
Multicollinearity + Relief-F	0.187	0.210	0.218	0.384	**0.030**	0.031
Grand Total	0.182	0.207	0.216	0.393	**0.027**	0.028

Similar to Random Forest and Subspace KNN ensembles, the CFS feature selection was the only approach that was statistically different among those considered and the least favourable feature selection approach. The feature selection approach that provided the highest kappa statistic was multi-collinearity removed in combination with further filtering using relief-F. The feature selection approach CFS did not perform as favourably as other SVM models.

The benchmark SVM classifiers (with ECOC in multi-classification) did not perform as well as the random forest ensemble model in terms of the kappa static. Although having a lower kappa value, unlike bagging, the SVMs were better able to differentiate between the common frequently occurring return class types and rarer event outcomes at the expense of misclassifying a portion of the common events. In other words, the SVM classifiers were more likely to predict the rarer class outcomes than bagging.

The use of a linear kernel within the SVM was sufficient when the number of features was larger than the number of observations. Most trading systems generated a training set containing a number of events less than the number of candidate features considered. The greater number of available candidate features compared to training events allowed the SVM to find the support vectors that reflected the division among the different event outcomes without the use of more complex kernelling methods.

Proposed Portfolio Strategy

The following section discusses the proposed strategy resulting from this research. The proposed strategy takes into consideration the interaction among the controllable factors of a technical trader and attempts to quantify the competitive advantages offered.

Addressing Significant Factors shown in the ANOVA

In line with commonly accepted practices in finances, this research proposes that technical traders adopt trading strategies that involve monitoring a large number of financial securities within a portfolio. This approach will help to diversify the risk of performance associated with any one particular financial security.

The classification models appear to provide the greatest advantage when combined with trading systems representing the intermediate trend and less sensitive to market noise. In the context of this research, the trading system SMA2 (20 fast and 5 slow) offered the greatest competitive advantage among the trading systems considered based on the kappa statistic.

As shown in *Figure 37*, there is a strong positive relationship (rho = 0.73) between the event return and the event duration for events generated with the trading system SMA2(20,5), suggesting this specific trading system is better able to identify price movements that are both directional and sustained compared to the other trading systems considered in this research. The longer time period for the SMA2 trading system smooths out daily market fluctuations leading to fewer, but more significant events that were likely to represent greater changes in behaviour and equilibrium.

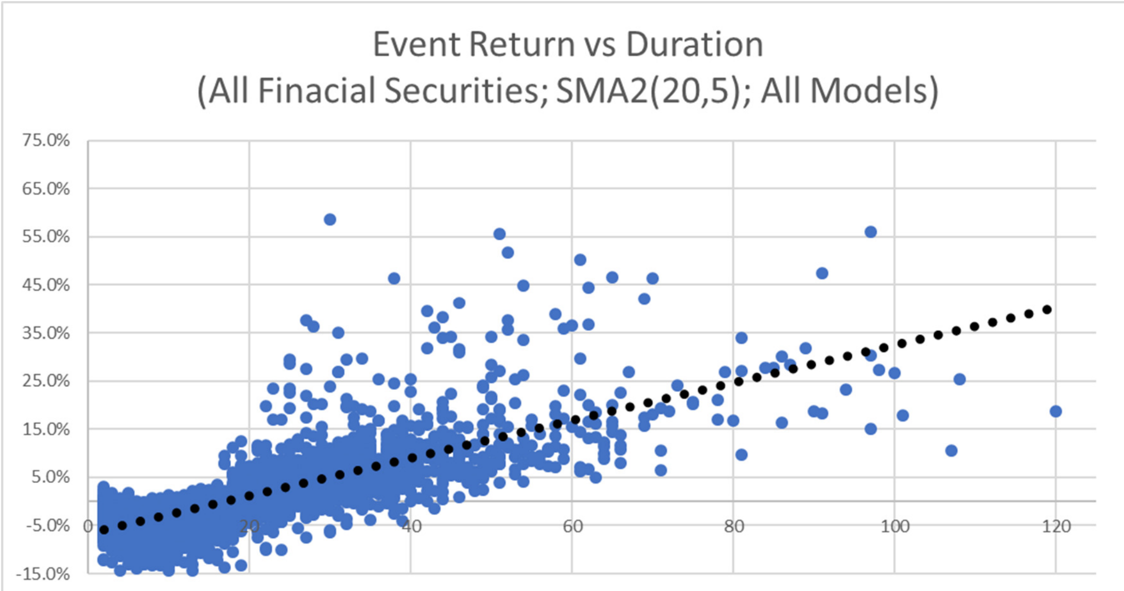


Figure 37. Duration (in days) versus return for SMA2(20,5).

In regards to the classification models, Table 32 presents a summary of the key performance indicators of the most favourable model among each ensemble type. As shown in Table 32, the bagging random forest ensemble shows the greatest competitive advantage for the SMA2(20,5) trading system based on the kappa static.

Table 32

Two Simple Moving Average (20,5) Performance Metrics

All Securities SMA(20,5) only By Model	Average of Macro F–score	Average of Macro Recall	Average of Macro Precision	Average of Overall Accuracy	**Average of Kappa**	Average of MCC
Bagging: Random Forest All Features	0.186	0.224	0.259	0.422	**0.080**	0.102
Subspace: K-Nearest Neighbour All Features	0.110	0.165	0.173	0.197	**0.014**	0.014
Boosting Random under Sampling All Features	0.171	0.213	0.238	0.434	**0.047**	0.059
Support Vector Machine ECOC Multicollinearity + Relief-F	0.198	0.221	0.236	0.381	**0.062**	0.065

Note. ECOC = Error Correcting Output Code; MCC = Mathews Correlation Coefficient; SMA = Simple Moving Average.

Suggested strategy

Most ensemble models have an objective to minimize the overall error rate of the classifier, which tends to give focus to predicting the majority classes, leading to poor accuracy for the minority classes. The proposed strategy suggests using a classification model to filter out the majority of event opportunities that are anticipated to be within the less favourable majority classes, while taking a position in remaining classes. Table 33 shows the number of events associated with a predicted outcome label based on the trading system SMA2(20,5). Table 34 shows the weighted average event return for the predicted outcome.

Table 33

Count of Portfolio Events by Predicted Outcome and Model

Select Model	Count of **Predicted** Events by Outcome Class (True Positive in Brackets)						Total
	Trbl	Undesr	Unfav	Fav	Desr	Excl	
Bagging: Random Forest All Features	690 (373)	2,052 (830)	510 (182)	44 (9)	2 (0)	0 (0)	3,298
Boosting: Random under Sampling All Features	317 (102)	1,060 (374)	493 (122)	251 (27)	417 (16)	760 (14)	3,298
Subspace: K-Nearest Neighbour All Features	1,503 (689)	1,384 (598)	359 (134)	44 (10)	7 (0)	1 (0)	3,298
Support Vector Machine ECOC Multicollinearity + Relief-F	1,213 (527)	1,234 (521)	567 (169)	184 (22)	83 (11)	17 (3)	3,298
Actuals	*(956)*	*(1103)*	*(683)*	*(331)*	*(160)*	*(65)*	*(3,298)*

Note. ECOC = Error Correcting Output Code; SMA = Simple Moving Average; Trbl = Terrible; Undesr = Undesirable; Unfav = Unfavourable; Fav = Favourable; Desr = Desirable; Excl = Excellent

Although the TP rate shown in Table 33 is not as promising as desired, the models were helpful in terms of directional price movements which can be supported by the weighted average return of the predicted classes. Table 34 shows how the average return of the terrible and undesirable *predicted* event outcomes are less than the average return of all events, and conversely, the predicted outcome classes of unfavourable, favourable, desirable, and excellent are, in most cases, greater than the average of all events. This observation is offered in support of the position that classification models are able to generate a degree of competitive advantage over other technical traders that trade all trading signals of the trading system SMA(20,5).

Table 34

Average Events Return by Predicted Outcome and Model

Average of All Securities Trading System: SMA(20,5) by Select Model	Average Return of Predicted Outcome Class						
	Trbl	Undesr	Unfav	Fav	Desr	Excl	Average
Bagging: Random Forest All Features	0.50%	0.51%	0.92%	3.62%	27.41%	0.00%	0.63%
Boosting: Random under Sampling All Features	0.51%	0.58%	0.57%	0.79%	1.06%	0.50%	0.63%
Subspace: K-Nearest Neighbour All Features	0.62%	0.53%	0.67%	2.09%	9.51%	7.88%	0.63%
Support Vector Machine ECOC Multicollinearity + Relief-F	0.62%	0.21%	1.23%	0.23%	2.79%	5.06%	0.63%
Actuals	*-4.55%*	*-1.65%*	*2.46%*	*8.48%*	*14.26%*	*22.76%*	*0.63%*

Note. ECOC = Error Correcting Output Code; SMA = Simple Moving Average; ; Trbl = Terrible; Undesr = Undesirable; Unfav = Unfavourable; Fav = Favourable; Desr = Desirable; Excl = Excellent

The following Table 35 shows if a technical trader used a classification model to filter out events among all 90 securities in the verification set that predicted a class outcome label of terrible or undesirable, then the number of remaining events would be significantly less, but the average return per event would increase.

Table 35

Weighted Returns of Portfolio strategy, by Model Type

Proposed Portfolio Strategy Models for consideration	Number of Event Positions	Weighted Average Return
Bagging: Random Forest All Features	556	1.23%
Boosting: Random under Sampling All Features	1921	0.68%
Subspace: K-Nearest Neighbour All Features	411	0.99%
Support Vector Machine ECOC Multicollinearity + Relief-F	851	1.24%
Actuals – all events	*3298*	*0.63%*

Note. ECOC = Error Correcting Output Code.

This research work developed a proposed strategy that consists of a large portfolio of financial securities combined with a trading system that is less sensitive to market noise. In the case of this research, the verification portfolio consisted of 90 financial securities traded based on the trading signals of the SMA2(20,5) trading system. If the technical trader were to have traded all the trading signals, then the trader would have taken 3,298 positions and earned an average return of 0.63% per position. Using a bagging ensemble model to filter the trading opportunities based on anticipated event outcome, the trader would have taken 556 positions with an average return of 1.23%. If the trader wanted to take more positions, he or she could simply include more financial securities into the portfolio to be monitored for trading opportunities as defined by the trading system. Increasing the number of financial securities monitored is more favourable than increasing the sensitivity of the trading system (i.e., use a shorter time dimension).

CHAPTER VII: CONCLUSIONS

This research investigated the use of ensemble classification models to classify the anticipated outcome of an event defined by a technical trading system. The classification models used various technical indicators to measure aspects of the market sentiment in an effort to capitalize on hidden information regarding market behaviour contained within financial security market pricing.

The signals of a technical trading system were used to discretize the time series of a financial security into discrete events. The set of events associated with a financial security and trading system combination were grouped into outcome classifications. The most favourable approach for grouping events into outcome classes was clustering, using K-mean and multiple classes, specifically, $k = 6$. Different ensemble classification methods and an SVM classification model were then used in combination with measures of various aspects of market sentiment to anticipate the outcome class of new events that occurred in the evaluation data set.

The application of classification models allows technical traders to develop a broad, directional expectation of an event outcome defined by a technical trading strategy prior to entering into a position which results in more efficient capital allocation. This research work argues that risk is relative based on the predictability of event outcomes, or more specifically, the degree of alignment between the classification model anticipated outcome label and the actual event outcome label. The degree of competitive advantage offered by the classification models to a technical trader is proportional to the value of hidden information found in the time series data.

This research used an ANOVA experiment to demonstrate the harmonic mean of recall and precision, referred to as F-score (b_1), which varied statistically among the ensemble methods investigated. Furthermore, the ANOVA demonstrated a statistically significant interaction between the factors identified to contribute to the performance of the classification models, namely the selected trading system and financial security.

Technical traders do not have control over the price movements of a financial security, but they can search for a competitive advantage through the interactions of the trading system and the selected classification model. This research supports the premise that it is possible to develop an ensemble classification model that is able to predict the outcome of market events using a diverse set of technical market indicators with a level of performance superior to random chance, or phased differently, better than nothing. The degree of competitive advantage is best described as slight to moderate.

The trading system and classification model that offered the greatest combined competitive advantage based on the kappa value from the perspective of classifying outcome labels was the SMA2(20,5) trading system in combination with a bagging random forest ensemble using all features. Although the random forest ensemble was the most favourable, the performance of the SVMs benchmark using the feature set that removed significant multicollinearity, then further filtered the candidate features using relief-F also showed favourable performance characteristics. The SVM models were better able to classify the more rare and profitable events than the random forest ensemble. In contrast, the random forest ensemble was able to classify more event labels correctly, but without consideration for the meaning of the cluster label.

Although the performance of the ensemble methods statistically varied, the feature selection approach generally did not have a material impact on the F-score or kappa values within a particular ensemble method, with the exception of the CFS approach. The CFS feature selection technique consistently selected a small number of candidate features into the selected feature set. Therefore, it is presumed the CFS feature selection approach did not contain enough selected features to create diversity among the ensemble members, nor were there sufficient measures of market sentiment to capture the properties of time series data.

The recommended strategy includes adding large number of financial securities to the portfolio and monitoring for opportunities that are classified as other than the common, less significant events, therefore focusing on opportunities more likely associated with changes in equilibrium. Although this research demonstrates it is possible to provide a statistically significant advantage greater than random chance, the degree of advantage is modest at best. To improve model performance, additional features from data sources different than financial time series data are likely required.

Future Research

Shorter term trading: Intra-day timeframe

This research assumes that shorter periodicity data is likely to be more reflective of how behaviour driven the market becomes. Therefore, it may be worth applying this research to smaller bar size data (i.e., hourly periodicity data or other intra-day trading). The notion of the market moving to behavioural patterns may be more applicable in a shorter-term context.

Focus on modelling specific market events

It may be possible to develop a snapshot of data that could be used to model particular event types. For example, create a model of a pharmaceutical company that is releasing a new product. The developed application could be used to create a generic profile using several other pharmaceutical companies to create a collection of events based on the time period just prior to the release of a new product. This snapshot may be more reflective of the market behaviour towards this particular type of financial security experiencing a common market event.

Expand the features to include more than financial security time series data

Wang et al. (2018) proposed an approach that uses both social media and market technical indicators in stock market prediction. Including other data besides financial security pricing would likely introduce further measures and different aspects of market sentiment.

REFERENCES

Achelis, S. B. (2001). *Technical analysis from A to Z* (2nd ed.). New York, NY: McGraw Hill.

Aggarwal, C. C., Hinneburg, A., & Keim, D. A. (2001). *On the surprising behavior of distance metrics in high dimensional space*. Paper presented at the International Conference on Database Theory. Retrieved from https://bib.dbvis.de/uploadedFiles/155.pdf

Ahmed, N., Atiya, A., Gayar, N., & El-Shishiny, H. (2010). An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews, 29*(5/6), 594–621. https://dx.doi.org/10.1080/07474938.2010.481556

Aly, M., & Atiya A. (2006). Novel methods for the feature subset ensembles approach. *International Journal of Artificial Intelligence and Machine Learning, 6*(4), 1–7.

Anderson, D. R., Sweeney, D. J., & Williams, T. A. (2005). *Statistics for business and economics* (9th ed.). Mason, OH: Thompson Learning.

Bodie, Z., Kane, A., & Marcus, A. J. (2003). *Essentials of investments* (5th ed.). New York, NY: McGraw-Hill.

Breiman, L. (1996). Bagging predictors. *Machine Learning, 24*(2), 123–140. https://dx.doi.org/10.1023/A:1018054314350

Breiman, L. (2001). Random forests. *Journal of Machine Learning, 45*(1), 5–32. https://dx.doi.org/10.1023/A:1010933404324

Brofos, J. (2014, April). *Ensemble committees for stock return classification and prediction.* Retrieved from https://arxiv.org/pdf/1404.1492.pdf

CFA Institute. (2008). *CFA program curriculum: Level 1* (Vol. 5). Hoboken, NJ: Wiley.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16,* 321–357. https://doi.org/10.1613/jair.953

Chen, C., Liaw, A., & Breiman, L. (2004). *Using random forest to learn imbalanced data* (Report No. 666). Retrieved from http://statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf

Dash, M., & Liu, H. (1997). Feature selection for classification. *Intelligent Data Analysis, 1*(1–4), 131–156. https://doi.org/10.1016/S1088-467X(97)00008-5

Dash, R., & Dash, P. K. (2016). A hybrid stock trading framework integrating technical analysis with machine learning techniques. *Journal of Finance and Data Science, 2*(1), 42–57. https://dx.doi.org/10.1016/j.jfds.2016.03.002

Diebold, F. (2004). *Elements of forecasting* (3rd ed.). Mason, OH: Thompson.

Fang, Y., & Xu, D. (2003). The predictability of asset returns: an approach combining technical analysis and time series forecasts. *International Journal of Forecasting, 19*(3), 369–385. https://dx.doi.org/10.1016/S0169-2070(02)00013-4

Fletcher, T. (2008, December). *Support vector machines explained.* Retrieved from https://cling.csd.uwo.ca/cs860/papers/SVM_Explained.pdf

Freund, Y., & Schapire, R. E. (1996, July). Experiments with a new boosting algorithm. *Proceedings of the Thirteenth International Conference on Machine Learning*, 148–156.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139. https://doi.org/10.1006/jcss.1997.1504

Goedhart, M., Koller, T. & Wessels, D. (2005). Do fundamentals- or emotions - drive the stock market? The McKinsey Quarterly 2005 special edition: Value and Performance. http://kwu-alumni.org/moto/media/5559046187646.pdf

Gold, S. C., & Lebowitz, P. (1999). Computerized stock screening rules for portfolio selection. *Financial Services Review*, *8*(2), 61–70.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research, 3*(7/8), 1157–1182.

Hall, M. A. (1999, April). *Correlation-based feature subset selection for machine learning*. Retrieved from https://www.cs.waikato.ac.nz/~mhall/thesis.pdf

Han, J., Kamber, M., & Pei, J. (2012). *Data mining–concepts and techniques* (3rd ed.). Waltham, MA: Morgan Kaufmann.

Hastie, T., Tibshirani, R., & Friedman, J. (2011). The elements of statistical learning: Data mining, inference, and prediction (2nd ed.). New York, NY: Springer.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 20*(8), 832–844. https://dx.doi.org/10.1109/34.709601

Hssina, B., Merbouha, A., Ezzikouri, H., & Erritali, M. (2014). A comparative study of decision tree ID3 and C4.5 [Special issue]. *International Journal of Advanced Computer Science and Applications, 10,* 13–19.

Hsu, C.-W., Chang, C.-C., & Lin, C.-J. (2016, May). *A practical guide to support vector classification*. Retrieved from https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

Imandoust, S. B., & Bolandraftar, M. (2013). Application of K-nearest neighbor (KNN) approach for predicting economic events: Theoretical background. *S B Imandoust et al. International Journal of Engineering Research and Application, 3*(5), 605–610.

John, G. H., Kohavi, R., & Pfleger, K. (1994, July). Irrelevant features and the subset selection problem. *Proceedings of the Eleventh International Conference on Machine Learning.* 121–129.

Jurman, G., Riccadonna, S., & Furlanello, C. (2012). A comparison of MCC and CEN error measures in multi-class prediction. *PLOS One, 7*(8). http://doi.org/10.1371/journal.pone.0041882

Kaufman, P. (2013). *Trading systems and methods* (5th ed.). Hoboken, NJ. John Wiley & Sons.

Kononenko, I., Šimec, E., & Robnik-Šikonja, M. (1997). Overcoming the myopia of inductive learning algorithms with RELIEFF, *Applied Intelligence, 7*(1), 39–55. https://dx.doi.org/10.1023/A:1008280620621

Krollner, B., Vanstone, B., & Finnie, G. (2010, April). *Financial time series forecasting with machine learning techniques: A survey.* Paper presented at the European Symposium on Artificial Neural Networks – Computational Intelligence and Machine Learning, Bruges Belgium.

Kuhn, M., & Johnson, K. (2013). *Applied predictive modeling*. New York, NY: Springer.

Landis JR, Koch GG. (1977). The measurement of observer agreement for categorical data. *Biometrics*, 33:159–174.

Leung, C. K.-S., MacKinnon, R. K., & Wang, Y. (2014, July). A machine learning approach for stock price prediction. *ACM International Conference Proceeding Series: Proceedings of the 18th International Database Engineering and Applications Symposium, IDEAS*, 274–277. Retrieved from http://www.cs.umanitoba.ca/~ywang/papers/ideas14.pdf

Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. *R News, 2*(6), 18–22.

Mangram, M. E. (2013). A simplified perspective of the Markowitz portfolio theory. *Global Journal of Business Research*, *7*(1), 59–70.

Martinez-Munoz, G., & Suarez, A. (2007). Using boosting to prune bagging ensembles. *Pattern Recognition Letters*, *28*(1), 156–165. https://dx.doi.org/10.1016/j.patrec.2006.06.018

MathWorks®. (2014). *Statistics and machine learning toolbox (R2014a).* Retrieved from http://www.mathworks.com/help/stats/index.html

Opitz, D., & Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research, 11,* 169–198. https://dx.doi.org/10.1613/jair.614

Oracle. (2018). *MySQL* (Version 8.0.12). Retrieved from https://dev.mysql.com/downloads/mysql/

Preethi, G., & Santhi, B. (2012). Stock market forecasting techniques: A survey. *Journal of Theoretical and Applied Information Technology, 46*(1), 24–30.

Quandl. (n.d.). *Quandl.* Retrieved from https://www.quandl.com/s

Rockefeller, Barbara. (2004). Technical Analysis for Dummies. Hoboken, NJ. John Wiley & Sons.

Robnik-Šikonja, M., & Kononenko, I. (2003). Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning, 53*(1/2), 23–69. https://dx.doi.org/10.1023/A:1025667309714

Seiffert, C., Khoshgoftaar, T. M., Van Hulse, J., & Napolitano, A. (2008). *RUSBoost: Improving classification performance when training data is skewed.* Paper presented at the Proceedings – International Conference on Pattern Recognition (19th International Conference on Pattern Recognition), Boca Raton, FL.

Sezer, O. B., Ozbayoglu, M., & Dogdu, E. (2017). A deep neural-network based stock trading system based on evolutionary optimized technical analysis parameters. *Procedia Computer Science, 114*, 473–480. https://doi.org/10.1016/j.procs.2017.09.031

Sharpe, W., Alexander, G., & Bailey, J. (1999). *Investments* (6th ed.). Upper Saddle River, NJ: Prentice Hall.

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, *45*(4)*,* 427–437. https://dx.doi.org/10.1016/j.ipm.2009.03.002

Tay, F. E. H., & Cao, L. (2001). Application of support vector machines in financial time series forecasting. *International Journal of Management Science, Omega, 29,* 309–317. https://dx.doi.org/10.1016/S0305-0483(01)00026-3

Tsai, C.-F., & Hsiao, Y.-C. (2010). Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches. *Decision Support Systems, 50*(1), 258–269. https://doi.org/10.1016/j.dss.2010.08.028

Tsai, C.-F., Lin, Y.-C., Yen, D. C., Chen, Y.-M. (2011). Predicting stock returns by classifier ensembles, Applied Soft Computing, Volume 11, Issue 2, March 2011, Pages 2452-2459, ISSN 1568-4946

Tsai, C.-F., & Wang, S.-P. (2009). Stock price forecasting by hybrid machine learning techniques. *International Multiconference of Engineers & Computer Scientists, 1*, 755–760.

University of Waikato. (n.d.). *Weka 3: Data mining software in Java.* Retrieved from https://www.cs.waikato.ac.nz/ml/weka/

Van den Poel, D., Chesterman, C., Koppenm, M., & Ballings, M. (2016, July). *Equity price direction prediction for day trading: Ensemble classification using technical analysis indicators with interaction effects*. Paper presented in the 2016 IEEE Congress on Evolutionary Computation, Vancouver, Canada. https://dx.doi.org/10.1109/CEC.2016.7744227

Visual Paradigm. (2018). Visual paradigm (Version 15.0). Retrieved from https://www.visual-paradigm.com/download/archive/

Wang, Q., Xu, W., & Zheng, H. (2018). Combining the wisdom of crowds and technical analysis for financial market prediction using deep random subspace ensembles. *Neurocomputing*, *299*, 51–61. https://dx.doi.org/10.1016/j.neucom.2018.02.095

Weckman, G. R., & Agarwala, R. (2003, May). Identifying relative contribution of selected technical indicators in stock market prediction. *IIE Annual Conference Proceedings*, 1–6.

Weiss, G. M. (2004). Mining with rarity: A unifying framework. *SIGKDD Explorations, 6*(1), 7-19.

Witten, I., Frank, E., & Hall, M. A. (2011). *Data mining: Practical machine learning tools and techniques* (3rd ed.). Burlington, MA: Morgan Kaufmann.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Qiang, Y., Motoda, H., . . . Steinberg, D. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems, 14*(1), 1–37. https://dx.doi.org/10.1007/s10115-007-0114-2

Yoo, P. D., Kim, M. H., & Jan, T. (2005). Machine learning techniques and use of event information for stock market prediction: A survey and evaluation. *International Conference on Computational Intelligence for Modelling, Control and Automation,* 835-841.

APPENDIX A: GLOSSARY OF TERMS

Terms are grouped based on relationship rather than alphabetically.

Term	**Definition** (as used in this thesis)
Application	Refers to the developed MATLAB program (MathWorks®, 2014) as a whole and all other related technical implementation created to support this research.
Financial Security	A financial security is a tradable asset such as a debenture (bond), equity (stock), or derivative (option). A security is represented by a ticker symbol, listed on a securities exchange, and has publicly available pricing history. In this research, a financial security refers to a randomly selected stock on the S&P 1500.
Periodicity	Periodicity refers to the unit of frequency of a time series. For purposes of this research, the periodicity refers to the daily price bar that presents end-of-day (EOD) data. EOD fields include open, high, low, close, and volume.
(Technical) Trading System	A trading system refers to a technical trading system that is used to define the beginning and end of an Event. In this research, a trading system contains an entry trigger and an exit trigger.
Profile	A profile contains a financial security, a technical trading system, and a trading strategy (long or short) that are combined to generate a collection of events.
Event	An event is a dynamic cross-section of time defined as the period between the exit trigger of sequential events. An event can have a state of monitoring, active, or historic. An event contains a collection of candidate features. An event is spawn in a monitoring state. An entry trigger causes the event to transition from a monitoring state to an active state. The exit trigger causes the event to transition from an active state to a historic state, and a new event being spawn.
Event Outcome (Class)	The event outcome refers to the response variable Y of a model. The event outcome is a classification label (e.g., excellent, favourable, unfavourable, or terrible in the case of binning, or excellent, desirable, favourable, undesirable, etc., in the case of clustering).
Technical Indicator	A technical indictor is a metric derived from the pricing of a financial security (e.g., Relative Strength Index, William's %R, etc.).

Time Dimension	Time dimension refers to the time parameter of a technical indicator that specifies the number of days contained in the calculation. For example, an exponential moving average with a time dimension of n = 10 considers 10 data points (days) in the calculation.
Indicator Derivative	Derivative refers to a permutation of a technical indictor. The two derivatives are acceleration or velocity.
Candidate Feature	A candidate feature refers to a potential predictor variable of a model and is a time-dimensioned derivative of a technical indictor. A candidate feature observation occurs at the time the trading system generates an entry trigger and represents an aspect of the market sentiment.
Snapshot	A snapshot is a collection of the historic events. A snapshot is defined by a start date and an end date. Historic profile events are used to form the snapshot which in turn becomes the common data source for each of the classification models compared in this research.
(Snapshot) Tuple	A snapshot tuple contains the data of an event object in a flat record format. A snapshot tuple consists of the event outcomes (duration and return) and all candidate feature's observations.
Feature Set	A feature set is a subset of candidate features selected from the pool of available candidate features in the snapshot based on one of four feature selection approaches. The feature set becomes the input/predictors used to develop the classification models.
Classifier	A model is a supervised machine learning method that maps the selected features/predictors to an event outcome/class label. This research utilizes three common supervised learning algorithms to classify the outcome of an event, specifically (a) decision trees, (b) K-Nearest Neighbour, and (c) support vector machines. A classifier is sometimes referred to as an ensemble member, learner, or weak learner in this thesis.
Ensemble	An ensemble is a collection of classification models (classifiers) that combine decisions of the individual classifiers into a single prediction.
Model	The term model can refer to an ensemble of classifiers or an individual member of an ensemble, depending on the context.

References

MathWorks®. (2014). *Statistics and machine learning toolbox (R2014a).* Retrieved from http://www.mathworks.com/help/stats/index.html

APPENDIX B: TECHNICAL INDICATORS

(CANDIDATE FEATURES)

The ensemble classification models utilize technical indicators as candidate features (predictors). These technical indicators measure various aspects of market sentiment towards a financial security at the time an event entry signal occurs. The technical indictors have been organized by the researcher into aspects of measured market sentiment, namely trend, momentum, volatility, and volume. This appendix describes the specific technical indicators used as candidate features within this research.

Trend

A trend can be defined as the general direction (i.e. trending upward or downward) of the market for a financial security and can vary in length and intensity. Three technical indicators used to capture the characteristics of a trend in this research are the slope of a simple linear regression model, an exponential moving average relative to the closing price, and J. Welles Wilder Jr.'s Directional Movement.

Slope of Simple Linear Regression (b_1)

The general regression model is as follows:

$$y = \beta_0 + \beta_1 x + \varepsilon$$

The terms β_0 and $\beta_1$are the parameters of the model, and ε is a random variable referred to as the error term. The error term accounts for variability in *y* that cannot be explained by the linear relationship between *x* and *y* (Anderson, Sweeney, & Williams, 2005, p. 555).

To create an estimated trend projection the independent variable time (*t*) is substituted for *x*, and the trend value of the time series for the period *T* is substitute for the estimated response variable $\hat{y}$. The independent variable time (*t*) is equal to one (*t* = 1) for the first observation in the time series, t=2 for the time of the second observation, and so on. The value *t* = 1 represents the oldest observation and t=*n* equals the most recent observation (Anderson et al,. 2005, p. 795).

$$T_t = b_0 + b_1 t$$

Where

- T_t = the trend value of the time series in period t
- b_o = intercept of the trend line
- b_1 = slope of the trend line
- *t* = time

The formula for computing the estimated regression coefficients (b_1 and b_0) are as follows (Anderson et al., 2005, p. 795).

$$b_1 = \frac{\sum tY_t - (\sum t \sum Y_t)/n}{\sum t^2 - (\sum t^2)\,/n}$$

$$b_0 = \bar{Y} - b_1\bar{t}$$

Where

- Y_t = value of the time series in period t
- n = number of periods
- $\bar{Y}$= average value of the time series; that is, $\bar{Y} = \sum Y_t\,/n$
- $\bar{t}$= average value of t; that is, $\bar{t} = \sum t\,/n$

The slope (b_1) of the linear regression line, or angle at which the financial security price is rising or falling, is the component of the regression equation used as a candidate feature in this research. The slope shows how quickly prices are changing over a period of time and can be used as a proxy to measure the strength of the Trend. The sensitivity, or degree that the slope changes, is dependent on the Time Dimension, or number of observations considered. A longer calculation period (greater number of observations) will result in the regression line changing slowly and a small number of observations will result in the slope changing quickly.

Exponential Moving Average Close

An Exponential Moving Average (EMA) is a weighted average of past time series values. An EMA applies a weighting factor such that each older observation decreases in value exponentially resulting in more emphasis being placed on the recent observations. The formula for calculating the EMA is as follows (Anderson et al., 2005, p. 787).

$$EMA_t = \alpha \times Y_t + (1-\alpha) \times EMA_{t-1}$$

Where

- EMA_t = is the Exponential Moving Average of the time series for time t
- Y_t = actual observed value at time t
- EMA_{t-1}= is the Exponential Moving Average of the previous period
- α= is the smoothing constant (0<= α <=1)

The weight given to the current period observation Y_t is α, and the weight given to the previous exponential moving average is $(1-\alpha)$. Although the EMA is a weighted average of all past observations, all past data does not need to be saved and computed for the next period (Anderson et al., 2005, p. 787). The coefficient α represents the degree of weighting decrease, a constant smoothing factor between 0 and 1. Higher values of α discount older observations more rapidly. Most technical traders feel more comfortable working with time periods rather than percentages. Alpha (α) may be expressed in terms of *n* time periods with the formula α = 2/(*n*+1) Achelis (2001). For example, *n* = 19 is equivalent to α = 0.1.

As a candidate feature within this research, the EMA is then expressed as a percentage of the current day close

$$\frac{EMA_t - Close_t}{Close_t} \times 100$$

Directional Movement

The Average Directional Movement is a combination of the Positive Directional Indicator (abbreviated DI_t^+) and the Negative Directional Indicator (abbreviated DI_t^-), all of which were developed by J. Welles Wilder Jr. (1978, pp. 35–48).

To calculate DI_t^+ and DI_t^-, the positive and negative daily directional movement needs to be calculated first. The positive and negative Directional Movements (DM_t^+ and DM_t^-) are calculated separately and the value not used for the period is set to zero. For a daily periodicity, the calculations of the DM_t^+ and DM_t^- are as follows and illustrated in Figure B1 below.

- Up Movement = Today's High − Yesterday's High, $H_t - H_{t-1}$.
- Down Movement = Yesterday's Low − Today's Low, $L_t - L_{t-1}$.
- When an inside period occurs, both positive and negative Directional Movement are set to zero
- IF Up Movement > Down Movement and Up Movement > 0, THEN
    - DM_t^+ = Up Movement, and DM_t^+= 0
- IF Down Movement > Up Movement and Down Movement > 0, THEN
    - DM_t^- = Down Movement, and DM_t^- = 0

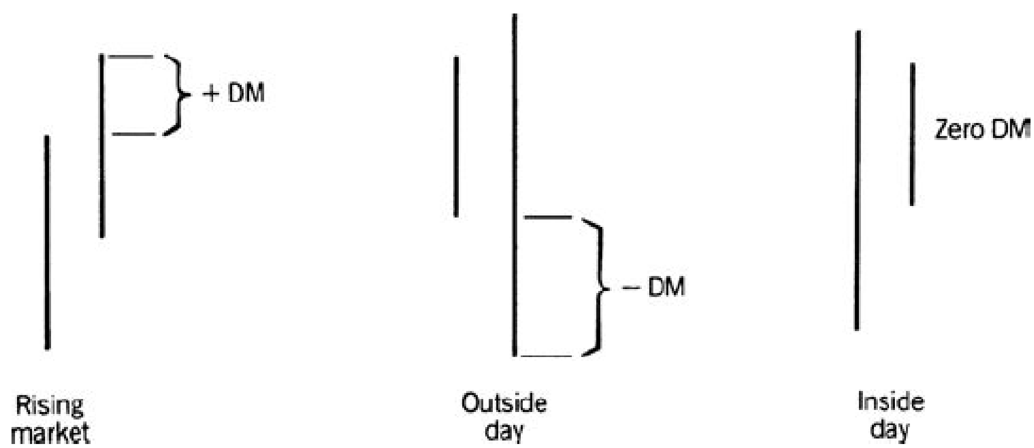


Figure B1. Directional movement.

Note. Based upon the works of Kaufman (2013, p. 1063) and Wilder, 1978, (pp. 35–48).

The daily directional movements are then summed over *n* days (represented by DM_n^+ and DM_n^-) to facilitate smoothed directional movement values as shown in the equations below.

Positive Directional Movement for *n* Periods

$$DM_n^+ = \sum_{i=1}^{n} DM_i^+$$

Negative Directional Movement for *n* Periods

$$DM_n^- = \sum_{i=1}^{n} DM_i^-$$

To determine the Directional Indicator, the daily true range (TR_t) needs to be calculated. The True Range is the maximum value of the following three differences:

- Today's high – Today's low ($H_t – L_t$)
- Today's high – Yesterday's closing ($H_t – C_{t-1}$)
- Today's closing – Yesterday's low ($C_t – L_{t-1}$)

The true range values are then summed over the same period of length n (TR_n)

Total Range for *n* Periods

$$TR_n = \sum_{i=1}^{n} TR_i$$

The directional indicator is the sum of the directional movement for n-periods divided by the sum of the true range for n-periods multiplied by 100. The notation "n" refers to the period over which the values are smoothed.

Positive Directional Indicator

$$DI_{t(n)}^{+} = \frac{DM_n^{+}}{TR_n} \times 100$$

Negative Directional Indicator

$$DI_{t(n)}^{-} = \frac{DM_n^{-}}{TR_n} \times 100$$

The Average Directional Movement indicator is a calculation based on $DI_{t(n)}^{+}$ and $DI_{t(n)}^{-}$(Equation below). When an upward trend is sustained, the $DI_{t(n)}^{-}$ value moves towards zero, and $DI_{t(n)}^{+}$ becomes larger. The ADX is normalized in order to express the value between 0 and 100 and is then multiplied by 100 to express the decimal percentage to a whole number. The absolute value prevents ADX from becoming negative.

$$\boldsymbol{ADX_t = \frac{\left| DI_{t(n)}^{+} - DI_{t(n)}^{-} \right|}{DI_{t(n)}^{+} + DI_{t(n)}^{-}} \times 100}$$

<u>Momentum</u>

Momentum measures the degree of change in a financial security pricing over a period of time. Momentum is an important concept as it provides insight on the behaviour of market participants. As an example, when the price of a security begins to rise, market participants tend to quickly acquire a position in the security, which in turn causes the price to increase at a faster rate. Simply put, demand creates demand. Conversely, once the momentum of the security fades, an increasing number of investors will opt to sell, resulting in a price drop. Three technical indicators are used to measure momentum in the research, namely the Relative Strength Index, Accumulation/Distribution Oscillator, and Williams's %R.

Relative Strength Index (RSI)

Developed by Welles Wilder Jr. (1978), the Relative Strength Index (RSI) is a momentum oscillator. The RSI computes momentum as a ratio of higher closes to lower closes in an effort to measure the internal strength of the security (Achelis, 2001). The RSI is expressed as a value ranging from 0 to 100 and is calculated as follows:

$$RSI = 100 - \left(\frac{100}{1 + RS}\right);\ RS = \left(\frac{AU}{AD}\right)$$

- AU = the average upward price movements during the past *n*-observations
- AD = the average downward price movements (stated as a positive numbers) during the past *n*-observations

Wilder (1978, p. 68) suggested the significant threshold levels for the RSI indicator are 30 and 70. The centre line for the Relative Strength Index is 50. If the relative strength index is below 50, the given security's losses are greater than the gains. When the relative strength index is above 50, the gains are greater than the losses over the previous *n*-observations.

[Average] Accumulation/Distribution (A/D) oscillator

In 1972, Jim Waters and Larry Williams (as cited in Kaufman, 2013) published a description of the Accumulation/Distribution (A/D) Oscillator. The oscillator defined by Waters and Williams uses a unique form of relative strength to measure the implied direction of the day's trading (Kaufman, 2013, p. 397). The A/D Oscillator is calculated as followed:

$$ADO_t = \frac{(H_t - O_t) + (C_t - L_t)}{2 \times (H_t - L_t)} \times 100$$

The maximum value of 100 is reached when a market opens (O) at the low (L) and closes (C) at the high (H). The Waters-Williams A/D Oscillator inherently adjusts to higher or lower trading ranges (volatility) as a result of the divisor being a multiple of the day's trading range (Kaufman, 2013, p. 397).

As a candidate feature within this research, the Accumulation/Distribution (A/D) is smoothed by taking the simple moving average:

$$Mean\ ADO_t = \frac{\sum_{i=n}^{n} ADO_i}{n}$$

Williams %R

Williams' %R is a momentum indicator that measures overbought/oversold levels by showing the current closing price in relation to the highs and lows of the past *n*-observations (Achelis, 2001). The purpose of the indicator is to convey whether a stock or commodity is trading near the high or the low of its recent trading range. Williams' %R is different from stochastic as it measures the strength of the market close compared to the high of the past *n*-periods (Kaufman, 2013, p. 401). As the close gets stronger the value of %R gets smaller. The William's %R indicator value can be calculated as followed:

$$\%R = \frac{Buying\ Power}{Range} = \frac{\max\limits_{t-n}(High) - Close_t}{\max\limits_{t-n}(High) - \max\limits_{t-n}(Low)} \times 100$$

<u>Volatility</u>

In finance, volatility is a common measure of risk. Volatility measures price instability without consideration to the direction of the general trend. The volatility of a given security, or the market as a whole, can change frequently or remain relatively constant depending on the market sentiment. The changing of volatility over time is more formally termed Heteroskedasticity.

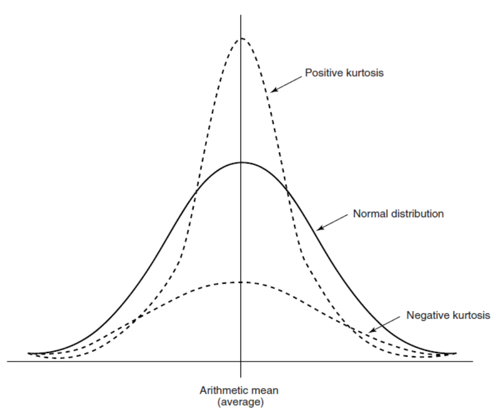
Market participants are concerned with volatility for a number of reasons. Firstly, the greater the variance in a security's price the greater the emotional stress associated with the investment. Secondly, volatility can lead to wider distribution of returns within a portfolio of financial securities. Lastly, volatility presents opportunities to buy securities low and sell high. When applying the principles of volatility to technical analysis, the more orderly the price data, the more reliable "trend" based indicators become. Conversely, highly volatile securities allow for opportunities of significant gains in a short period of time. As candidate features in this research, volatility is measured by monitoring changes in Kurtosis over time, J. Welles Wilder Jr's Volatility Index, and Efficiency ratio.

Excess Kurtosis

Kurtosis is the statistical measure that is used to identify when a distribution is more or less 'peaked' than a normal distribution (CFA Institute, 2008, p. 302). Frequency distributions are important as the standard deviation is not overly effective in describing skewed distributions, which are common for most price data (Kaufman, 2013, p. 43).

The calculation for kurtosis involves finding the average of the deviations from the mean raised to the fourth power and then standardizing the average by dividing the standard deviation raised to the forth power. For all normal distributions, kurtosis is equal to 3. Excess kurtosis is commonly used as it is easier to see abnormal distributions. Excess kurtosis is equal to the kurtosis minus three (CFA Institute, 2008, p. 302).

$$Excess\ K = \frac{\sum_{i=1}^{n}(p_i - \bar{P})^4}{(n-1)\sigma^4} - 3$$



The excess Kurtosis measurement is an unbiased assessment of whether prices are trending in a direction or moving sideways (relatively stable). When period returns move steadily higher, then the distribution will be flatter and cover a wider range resulting in negative excess kurtosis. If price returns are relatively stable, then the frequency will show grouping around the mean resulting in positive excess kurtosis. A positive kurtosis is typical of a sideways market and negative kurtosis (flatter distribution) is characteristic of a trending market (Kaufman, 2013, pp. 42–43).

Volatility Index

The traditional ‘range’ of a security’s price is simply the high minus low and is used as a measure of price volatility for the given period. Wilder’s (1978) True Range (p. 21) extends the definition of range to include the previous period’s closing price if it was outside of today's range.

Wilder defines the true range (TR) are the largest of the following:

- Today’s high – Today’s low ($H_t – L_t$)
- Today’s high – Yesterday’s closing ($H_t – C_{t-1}$)
- Today’s closing – Yesterday’s low ($C_t – L_{t-1}$)

These rules can be simplified into the following equation

$$true\ range\ (TR) = \max(high_t, low_t) - min(low_t, close_{t-1})$$

Large or increasing ranges suggest that investors are prepared to continue to bid up or sell down a security through the course of the period, while a decreasing range suggests market interest in the security is decreasing. The Volatility Index (VI_t) is a weighted average of the true range over *n* periods (Wilder, 1978, p. 22).

$$VI_t = \frac{(n \times VI_{t-1}) + TR_t}{n}$$

Efficiency Ratio

Market noise can be viewed as the erratic movement that surrounds the underlying direction of prices. There are a number of ways to measure noise including the Efficiency Ratio. The Efficiency Ratio is calculated by dividing the net price change by the sum of the individual period price changes within a timeframe, each taken as a positive number (Kaufman, 2013, p. 10). The following is the formula for the Efficiency Ratio

$$ER_t = \frac{|P_t - P_{t-n}|}{\sum_{t=t-n}^{i=t} |P_i - P_{i-1}|}$$

Where

- P_t = the most recent (current) closing price in the time series
- P_{t-n} = the oldest (first) closing price in the time series
- P_i = the security closing price at period *i*
- P_{i-1} = the previous security close price in relation to period *i*

Noise is market movement that has no significant direction or price movement. Generally, markets with a higher efficiency ratio are more favourable for trend following trading systems (Kaufman, 2013, pp. 1070–1072).

Volume

Volume, or trading volume, refers to the number of shares or contracts traded in a financial security or in an entire market during a given period of time. Volume is the measure of trader participation. When a rise in security price is accompanied by an increasing volume, it is possible to conclude the directional move is associated with market participation. Low volume levels within a security are usually the result of indecisive expectations among market participants, which typically occurs in a stable market. Two common technical indicators used to measure volume are the Exponential Moving Average and the Price and Volume Trend (PVT). A third indicator was developed based on ideas of Sibbett's Demand Index.

Exponential Moving Average of Volume

The Exponential Moving Average of Volume (*EMA Vol*) is derived using the same calculations as Exponential Moving Average of a financial security's closing price as previously described, except the period volume is substituted for the period close. As a candidate feature in this research, the EMA is expressed as a percentage of the current day volume.

$$\frac{EMA\_Vol_t - Volume_t}{Volume_t} \times 100$$

Price and Volume Trend (PVT)

The Price and Volume Trend (PVT) indicator is similar to the more common "On Balance Volume" (OBV) indicator. PVT is a cumulative total of volume that is adjusted depending on changes in closing prices. The PVT adds or subtracts a portion of the volume to the cumulative total based on the change in price relative to the period's close. The PVT adds a small portion of volume to the indicator when the price changes by a small percentage and adds a large portion of volume to the indicator when the price changes by a large percentage (Achelis, 2001).

The PVT is calculated by multiplying the period's volume by the percent that the security's price changed, and adding this value to a cumulative total (Achelis, 2001).

$$PVT_t = \left(\left(\frac{P_t - P_{t-1}}{P_{t-1}}\right) \times Volume\right) + PVT_{t-1}$$

In this research, the PVT is then expressed as a percentage of the current day volume.

$$\frac{PVT_t - Volume_t}{Volume_t} \times 100$$

Modified Sibbett's Demand Index

The Sibbett's Demand Index shows the difference between advancing and declining volume. The indicator shows the net flow of volume into or out of the market. The technique is similar to the approach used in Wilder's Relative Strength index (Kaufman, 2013, p. 548; Wilder 1978, p. 68).

$$Demand\ Index_t = \frac{\sum_{i=t-n+1}^{t} Upside\ Volume_i}{\sum_{i=t-n+1}^{t} Downside\ Volume_i}$$

Where

- Upside Volume = Total volume traded of securities that closed above their opening price
- Downside Volume = Total volume traded of securities that closed below their opening price

In this research, the Sibbett's Demand Index has been modified to the following

$$Modified\ DI = \left[\frac{\sum_{i=n}^{1}\left(\left(\frac{|P_i - P_{i-1}|}{P_{i-1}}\right) \times V_i\right)}{\sum_{i=n}^{1}\left(\left(\frac{|P_{i-1} - P_i|}{P_i}\right) \times V_i\right)} - 1\right] \times 100$$

Where

- P_n is the price at close at period *n*
- P_{n-1} is the price at close at period *n-1*

References

Achelis, S. B. (2001). *Technical analysis from A to Z* (2nd ed.). New York, NY: McGraw Hill.

Anderson, D. R., Sweeney, D. J., & Williams, T. A. (2005). *Statistics for business and economics* (9th ed.). Mason, OH: Thompson Learning.

CFA Institute. (2008). *CFA program curriculum: Level 1* (Vol. 5). Hoboken, NJ: Wiley.

Kaufman, P. (2013). *Trading systems and methods* (5th ed.). Hoboken, NJ. John Wiley & Sons.

Wilder, J. W., Jr. (1978). *New concepts in technical trading systems*. Alameda, CA: Hunter.